Desenvolupament d'un sistema adaptador d'interfícies d'usuari per a diferents discapacitats i dispositius

Víctor Llorens
Juanjo Pardo

Setembre 2010

Dedicat a

Dedicat als familiars, amics i tota aquella gent que ens ha donat suport durant aquests dos anys per a la realització d'aquest projecte.

En especial a les hores dedicades pels nostres tutors Montserrat Sendín i Juan Miguel López.

Gràcies.

Índex

Índex de figures

Capítol 1

Introducció

1.1 Motivació del projecte

En aquest món cada cop més informatitzat, any rere any augmenta el nombre d'usuaris d'aplicacions informàtiques, ja sigui a través del creixent mercat dels dispositius mòbils com del més tradicional però també creixent mercat de l'ordinador personal, ampliat recentment amb productes com netbooks o portàtils a preus impensables fa relativament poc temps.

Un percentatge d'aquests usuaris té algun tipus de discapacitat que afecta a la interacció amb les aplicacions.
Per contrapartida la gran majoria d'aplicacions es desenvolupen sense tindre aquest fet en compte, i en el cas que ho contemplin solen ser solucions fetes a mida per a un perfil d'usuari concret.

És cert que hi ha excepcions, però no és la norma general. La gran majoria de les aplicacions que hi ha actualment al mercat no té en compte les necessitats especials d'aquest conjunt d'usuaris. D'altra banda les solucions actuals per a aquest problema es centren a cobrir les necessitats de discapacitats concretes en comptes de proposar una solució d'àmbit més general.

1.2 Objectius del projecte

Es proposa implementar una solució genèrica, capaç de satisfer les necesitats de diferents perfils i de funcionar sota diferents dispositius, basada en modificar la interfície d'usuari de les aplicacions per tal de facilitar la interacció de l'usuari amb aquestes.

Això genera un seguit de problemes que s'han de tenir en compte abans de poder prendre cap decissió de disseny pel sistema:

- **Distingir entre usuari i usuari discapacitat**
  Abans de prendre cap decissió el sistema ha de saber d'alguna manera si l'usuari és discapacitat o no. És necessari ja que el sistema no ha de limitar-se a oferir canvis només per als usuaris discapacitats sinò que ha de poder millorar també l'experiència d'us de l'aplicació per a usuaris normals.

- **Múltiple ventall de discapacitats**
  Actualment estan reconegudes moltes discapacitats que impedeixen l'ús normal d'apliacions, i a més a més, el grau de discapacitat pot variar per segons quina discapacitat. Quan el sistema ha de tractar amb usuaris discapacitats aquest té que saber exàctament quin és el seu tipus de discapacitat i quin és el seu percentatge d'afectació.

- **Múltiple ventall de dispositius**
  En el mercat existeixen múltiples tipus de dispositius: ordenadors de sobretaula, terminals mòbils, PDA's, netbooks, tablets, electrodmèstics intel·ligents, i més. El sistema ha de ser capaç de distingir-los i poder adaptar qualsevol aplicació independentment del tipus de dispositiu, tenint en compte paràmetres com quin son els seus dispositius d'E/S, quines capacitats de connectivitat disposa, etc. A més a més cada un d'aquests dispositius, pot portar un sistema operatiu diferent que el sistema també necessita saber.

- **Múltiples tipus d'aplicacions**
  Les aplicacions poden estar implementades en múltiples tipus de llenguatges. Cada llenguatge té els seus avantatges i els seus inconvenients a l'hora de construir interfícies d'usuari i les funcionalitas que aquests ofereixen proporcionen una navegació diferent entre ells per a cada interfície. Estem parlant de events, elements de pantalla, menús, controls, i altres aspectes d'una interfície d'usuari

També es pretén que el sistema aprengui de les dades que pugui haber captat d'interaccions anteriors per tal d'oferir una modificació de la interfície d'usuari cada cop millor. Per tal de poder oferir aquesta funcionalitat existirà la posibilitat de que l'usuari de puntuar els últims canvis aplicats.

Els requisits principals que s'han tingut en compte a l'hora de desenvolupar aquest sistema per a assolir els seus objectius són els següents:

- Codificar les discapacitats d'usuari, creant un perfil d'usuari
- Codificar les peculiaritats del dispositiu, creant un perfil de dispositiu
- Proporcionar una solució capaç de convertir qualsevol aplicació Java en una aplicació adaptada
- Aquesta solució ha de poder monitoritzar la interacció de l'usuari amb l'aplicació i ha de ser conscient de l'estructura de la interfície d'usuari
- La solució ha de poder utilitzar les dades del punt anterior així com els dos perfils per determinar canvis a la interfície d'usuari que millorin la interacció
- L'usuari ha de tenir cert control sobre l'adaptació de la interfície d'usuari
- L'adaptació ha de ser transparent per a l'usuari. En conseqüència ha de ser ràpida

1.3 Estructura del document

Aquest document esta estructurat en 11 capítols, el primer dels quals és el que el lector llegeix, la introducció. A continuació es fa una breu descripció de cada un d'ells:

- **Capítol 2: Definició del problema**
  S'introdueixen els conceptes que hauran de ser resolts i s'acota la solució.
- **Capítol 3: Context d'accessibilitat**
  Es defineix el concepte d'accessibilitat i es mostra la situació actual quant a aquest tema.
- **Capítol 4: Java a l'actualitat**
  Una aproximació al món Java a diferents nivells. És un requisit esencial ja que es tracta del llenguatge amb que està implementada la solució i en el que estan implementades les aplicacions que es volen adaptar.
- **Capítol 5: Accessibilitat en Java**
  S'exposa les solucions que ofereix Java quant a accessibilitat.

- **Capítol 6: Estudi de les libreries d'Interfície d'Usuari a Java**
  Definició de les diferentes llibreries que utilitzen tant J2SE com J2ME per a oferir interfícies gràfiques i estudi de la capacitat d'adaptació de cadascuna.

- **Capítol 7: Solució proposada**
  Descripció a nivell de disseny de la solució que presenta aquest projecte.

- **Capítol 8: Tecnologies utilitzades i detalls de la implementació**
  Aprofundiment en les decisions de implementació i recull de les tecnologies que han sigut emprades en aquest projecte, siguin part de la solució final o no.

- **Capítol 9: Capacitat d'adaptació de la solució**
  Especificació de les adaptacions que és capaç d'oferir el sistema actualment.

- **Capítol 10: Casos d'estudi**
  Es verifica que el sistema funciona.

- **Capítol 11: Conclusions i treball futur**

- **Annex A: Preparació de l'entorn per al desenvolupament sobre dispositius mòbils (J2ME)**
  Com preparar l'entorn de desenvolupament per a J2ME pas a pas.

- **Annex B: Preparació de l'entorn per al desenvolupament d'escriptori (J2SE)**
  Com preparar l'entorn de desenvolupament per a J2SE pas a pas.

- **Annex C: Preparació de l'entorn per al desenvolupament del servidor**
  Com preparar l'entorn de desenvolupament per al servidor pas a pas.

Capítol 2

Definició del problema

La necesitat d'adaptar la interfície d'usuari a les necesitats d'aquest obre la porta a un nou concepte, anomenat *personalització*(REF) de la interfície d'usuari.

Tanmateix, l'objectiu d'aquest projecte no es limita només a les necessitats de l'usuari, a l'hora de millorar la interfície. Es parla, doncs, d'*adaptació* quan l'interficie d'usuari propicia una millor interacció amb l'usuari gràcies a canvis que neixen tant de les necesitats de l'usuari com d'altres paràmetres que no depenen d'aquest. Aquests paràmetres adicionals vénen marcats pel *context d'ús*[1]. Cal recordar, que degut a l'heterogeneïtat de l'entorn d'execució que es proposa, aquest context d'ús pot ser molt diferent d'un cas a un altre. Això és particularment quan es parla de dispositius mòbils, ja que l'entorn es caracteritza per canviar constantment (factors de lluminositat, soroll, capacitat d'atenció de l'usuari, pausa de l'ús de l'aplicació al rebre una trucada...).

2.1 Procés d'adaptació

Per tal de solucionar aquest problema l'aplicació inevitablement ha de passar per un procés d'adaptació. Aquest procés consta de varis passos atòmics que es defineixen a continuació i es concreten en el capítol *Solució proposada*.

1. **Captura de dades**

[1]Es defineix context d'ús com aquella informació útil que descriu una situació particular en la que un usuari realitza una tasca amb un sistema interactiu

Es recullen totes les dades que es pretenen utilitzar com a base per a deduir adaptacions.

2. **Modelat de dades**
   S'estructuren les dades recollides al punt anterior, de forma que un sistema informàtic en pugui fer ús.

3. **Inferència de l'adaptació**
   Mitjançant totes aquestes dades es genera una adaptació a partir de la inferència de les mateixes amb certes condicions.

4. **Materialització de l'adaptació**
   L'adaptació inferida s'aplica.

En la figura 2.1 es pot visualitzar tot el procés mencionat.

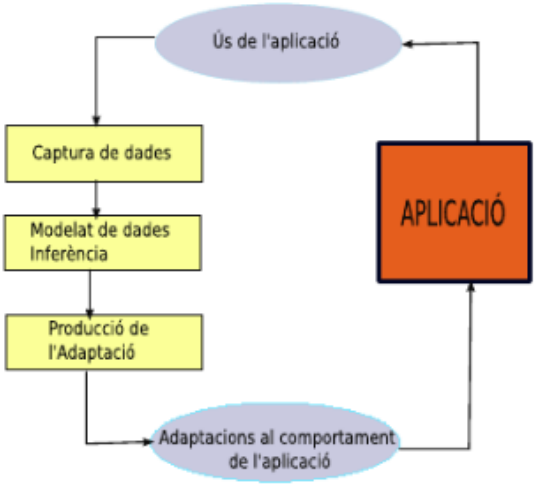

Figura 2.1: Procés d'adaptació

2.2 Tipus d'adaptacions

Les adaptacions es divideixen en dues famílies:

- **Adaptacions estàtiques**
  On l'usuari estableix d'avantmà els paràmetres d'adaptació, normalment definint un perfil d'usuari. Aquest concepte és conegut com a

adaptabilitat (REF), quan l'aplicació només té en compte el perfil d'usuari.
Un exemple clar d'adaptabilitat es trova a algunes aplicacions Web 2.0, com ara el portal *iGoogle*[2], *Gmail*[3] ó *Netvibes*[4].

- **Adaptacions dinàmiques**
  A les adaptacions dinàmiques és el propi sistema qui observa el patró de comportament de l'usuari mitjançant la monitorització de les seves accions, i va inferint quins canvis realitzar a la interfície progresivament. Quan aquest tipus d'adaptacions només té en compte l'usuari rep el nom de *adaptativitat* (REF).
  L'adaptativitat s'utilitza en la *hipermèdia adaptativa*[5], on alguns dels seus representants són el camps com el *e-learning* o el *e-bussiness*. Un exemple de cadascun d'ells el podem trobar a *InterBook*[6] i *Amazon*[7], respectivament.

2.2.1 Plasticitat de la interfície d'usuari

La necesitat d'adaptar la interfície d'usuari a un ampli ventall de factors, entre ells, els derivats de la mobilitat, sense descuidar els aspectes d'usabilitat porta a un nou concepte, anomenat *plasticitat*[1].

Es defineix plasticitat com la capacitat d'un sistema interactiu per suportar variacions en el context d'ús de manera econòmica i potencialment ergonòmica, tot preservant un conjunt definit de propietats d'usabilitat.

La plasticitat representa la capacitat d'una interfície d'usuari per variar en pro de la facilitat d'ús durant el temps.

Una interfície d'usuari pot esdevindre plàstica de dues formes diferent:

- Sent el propi dispositiu qui, d'alguna manera, sigui capaç de variar en el temps i de forma automàtica la interfície d'usuari. Aquest tipus de plasticitat és conegut com a *plasticitat implícita*.

[2]http://www.google.es/ig - Portal personalitzable de Google
[3]www.gmail.com - Aplicació web de correu, altament personalitzable
[4]www.netvibes.com - Lector web enriquit de feeds, completament configurable
[5]La hipermèdia adaptativa ofereix a l'usuari de la web 2.0 enllaços en els que l'usuari està potencialment interesat per sobre de la resta
[6]http://www.sis.pitt.edu/ peterb/InterBook.html - InterBook és una eina d'ajut a la creació de llibres electrònics adaptatius
[7]www.amazon.com - Tenda online. Exemple de web adaptativa

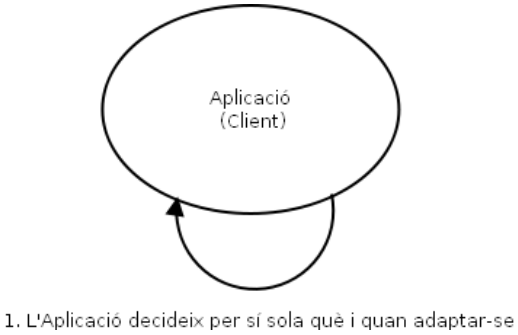

Figura 2.2: Plasticitat implícita

- Delegant l'adaptació a un sistema exterior, que després retornarà el contingut adaptat al dispositiu original, per tal de presentar-lo a l'usuari. Aquest tipus de plasticitat és conegut com a *plasticitat explícita*.

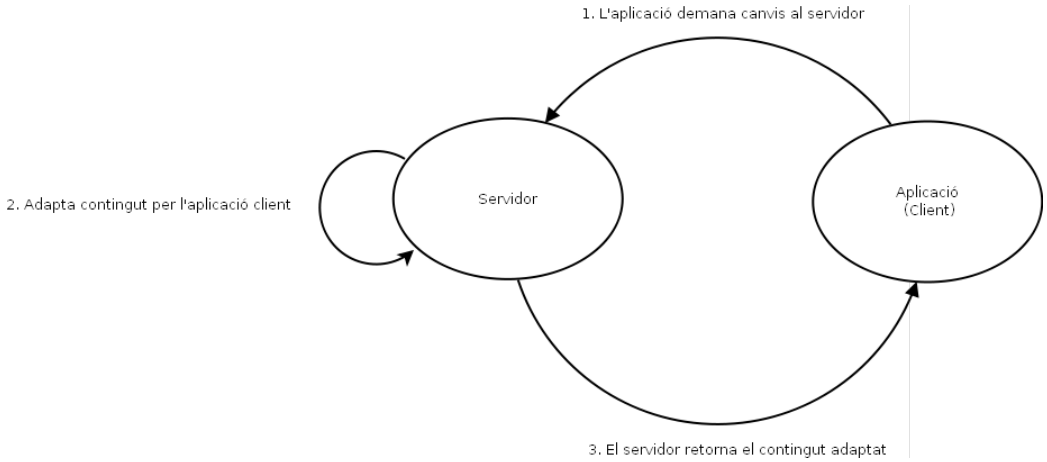

Figura 2.3: Plasticitat explícita

Els dos tipus de plasticitat no són mutuament excloients, sinó que pot donar-se el cas en el que s'apliquin els dols simultàniament. En aquest cas la plasticitat implícita s'encarregaria dels canvis més superficials, mentres que la plasticitat explícita abordaria els canvis més profunds, com ara una reestructuració de la IU, o qualsevol altra adaptació complexa. Combinar els dos tipus de plasticitats és anomenat *Visió Dicotómica de plasticitat*.

2.3 Enfocament escollit

Ja que la solució que es proposa és de caire genèric, on poden ser necesàries adaptacions molt diverses que haurien de ser calculades per dispositius que poden o no tindre una potència de còmput adient, s'ha optat per un enfocament basat en la plasticitat explícita, delegant la tasca de determinar l'adaptació més adient a un servei extern al dispositiu.

Quant a les possibles adaptacions, s'ha exposat que aquestes dependràn no només del perfil d'usuari, sinó que també tindràn a la seva disposició tant un perfil de dispositiu com dades sobre interaccions anteriors. Degut a que la solució necesita de l'ajut humà per a proveir el sistema amb els perfils necesaris però el sistema és a l'hora capaç d'oferir adaptacions de forma automàtica, es dedueix que les adaptacions que proporciona són a l'hora de tipus estàtic i dinàmic, sent el component dinàmic el prevalent.

Des del punt de vista de les aplicacions, les que generarà la solució seran aplicacions adaptables, ja que requereixen del perfil d'usuari per ser generades. No obstant un cop generades rebran propostes de canvi a la interfície d'usuari inferides externament a través, en part, de la monitorització de la interacció i que seran aplicades en temps d'execució sense l'intervenció de l'usuari. També es poden considerar, per tant, adaptatives.

Capítol 3

Context d'accessibilitat

3.1 Introducció a l'accessibilitat

En arquitectura, que un lloc sigui accessible vol dir que ha d'estar disponible per a totes les persones independentment de les condicions de l'entorn i de les discapacitats de qui en fa ús.

En termes d'informàtica això es tradueix en fer les aplicacions tant web com d'escriptori de manera que qualsevol persona pugui interactuar amb aquesta, tot això, independenment aquest cop del hardware i sistema operatiu on s'executa l'aplicació. La disciplina que s'ocupa d'aquest tema és l'IPO, Interacció Persona Ordenador.

Per a poder-ho dur a terme han sorgit els estàndards, per a poder fer les aplicacions segons un patró establert, fruit generalment d'un conveni com pot ser entre empreses, usuaris, associacions, institucions públiques, etc.

Actualment l'intent de fer les aplicacions accessibles el veiem, en major o menor mesura, en tots els dispositius: des d'ordenadors de sobretaula a dispositius mòbils passant per ordenadors portàtils o electrodomèstics.

3.2 Legislació i Pautes d'accessibilitat

Per a poder regularitzar l'accessibilitat doncs han tingut que sorgir estàndards, però amb aquests no n'hi ha prou. És una necessitat del ciutadà disposar d'aplicacions accessibles i en conseqüència, del sistema. És a dir, és l'estat qui ha de resoldre lleis que ajudin a la creació de llocs accessibles, que promo-

gui l'accessibilitat als seus propis llocs i que penalitzi a qui no les compleixi.

També hi ha legislació internacional i com s'ha dit abans, pautes creades per la comunitat.

3.2.1 Lleis espanyoles

Les primeres lleis en quant a accessibilitat resoltes per l'estat espanyol no es remunten més enllà de l'actual constitució, la del 78 (estem al 2010), són però articles molt antics en que la usabilitat informàtica no hi té massa cabuda. Així doncs als anys 80, quan els ordenadors de sobretaula ja començaven a estar presents en la societat va sorgir la necessitat d'establir noves lleis per sobre de la constitució atenuant aquesta mancança. S'enumeren a continuació:

- Llei 13/1982. D'integració social dels minusvàlids
- Llei 34/2002. Dedicada als serveis públics de la Societat de l'Informació i del Comerç Electrònic
- Llei 51/2003. D'Igualtat d'Oportunitats, No Discriminació i Accessibilitat Universal de les persones amb discapacitats[1]
- Reial Decret 1414/2006. Estableix els criteris de consideració per a persones amb discapacitat.
- Reial Decret 1417/2006. Regula els mecanismes per a queixes i reclamacions.
- Reial Decret 366/2007. En la que s'estableix un compromís que garanteix l'accessibilitat als ciutadans amb l'administració del Estat.
- Reial Decret 1494/2007 . Normalitza les condicions bàsiques d'accés i afecta entre d'altres a les AAPP i als prestadors de serveis de la Societat de la Informació. Per dur-ho a terme utilitza el compliment de les pautes Web Content Accessibility Guidelenes, explicades en el capítol següent.
- La Llei 49/2007. En la que s'estableix el régim d'infraccions i sancions en materia de igualtat d'oportunitats, no discriminació i accessibilitat universal de les persones amb discapacitat. Es jerarquitzen les infraccions en tres tipus: lleus, greus i molt greus.

[1]Del BOE - http://www.boe.es/boe/dias/2003/12/03/pdfs/A43187-43195.pdf

- La Llei 56/2007. De mesures d'Impuls de la Societat de l'Informació.

3.2.2 Lleis internacionals

Fora d'Espanya i a nivell mundial també existeixen lleis que regulen i estableixen certes pautes a seguir, com a mínim pels estats firmants, com oferir els serveis de la informació per als discapacitats. A continuació se'n llisten les més importants.

- D'àmbit mundial existeix la convenció de les Nacions Unides sobre el dret de les persones amb discapacitat[2]. Establerta el 30/març/2007. En aquesta s'adopten mesures per a regularitzar l'acces a les tecnologies de la informació i comunicació.

Figura 3.1: Alfabet braile

- D'àmbit europeu existeix el Dictamen del Comité Econòmic i Social Europeu sobre la legislació pel que fa l'accessibilitat electrònica, del 30/Maig/2007, en el que es proposa impulsar per a cada estat membre la legislació esmentada. Alguns dels detalls més importants d'aquesta legislació:
  - Promou estandaritzar els medis en que cada estat membre afronta l'accessibilitat establint uns estandards, otargant doncs als serveis ofertats d'interpolabilitat.
  - Directiva 2004/18/CE del 31/Març/2004 que obliga a tots els contractes públics de la societat de la informació cumpleixin certs criteris d'accessibilitat a partir del 2006. Avui en dia per tant ja s'està aplicant.
- A banda de les lleis existeixen altres tractats, plans o iniciatives com ara: Plan Info XXI, Iniciativa eEurope, Pla d'acció eEuropa 2002 i més.

[2]enable - http://www.un.org/spanish/disabilities/

Figura 3.2: Alfabet braile

Cada país però intenta basar-se en les pautes del WAI, ésent doncs un factor unificador i favorable per l'accessibilitat.

3.2.3 Web Accessibility Initiative

WAI (*Web Accessibility Initiative*[3]) forma part del consorci W3C. La seva finalitat és treballar amb organitzacions ja siguin públiques o privades d'arreu del mon per a desenvolupar estratègies, pautes, i recursos per ajudar a fer la web més accessible a les persones discapacitades.

Té una estructura d'organització que permet que qualsevol persona pugui aportar treball, diners o qualsevol tipus de recurs a l'organització. Sempre supervisats alguna d'aquestes tasques pot ser la revisió de documents publicats, l'implementació d'alguna funcionalitat o promocionant els seus productes. També tenen grups de treball.

Les pautes del WAI són referència a les lleis dels països, com ja hem mencionat anteriorment.

3.3 Tipus de discapacitats

De discapacitats n'hi ha de molts tipus, però a continuació es mencionen les que apareixen a les pautes del WAI, entre les quals hi ha les que s'ha basat aquest projecte.[3]

3.3.1 Visuals

Les discapacitats visuals es divideixen en 3 tipus:

[3]WAI - http://www.w3.org/WAI/

- **Ceguera**
  La ceguera suposa una reducció substancial o total de la visibilitat. Totalment irrecuperable.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Imatges que no tenen text alternatiu
  - Imatges complexes (e.g., gràfics o taules) que no estan descrites adequadament
  - Vídeo que no està transformat a text o àudio
  - Taules que no tenen sentit quan són llegides de manera lineal
  - Frames que no tenen l'alternativa "NOFRAME", o que no disposen de noms adequats a la informació a mostrar
  - Formularis que no poden ser seleccionats en una seqüència lògica o que estan mal etiquetats
  - Navegadors o eines que no tenen suport per a executar totes les comandes des del teclat
  - L'ús de formats de documents no estadard

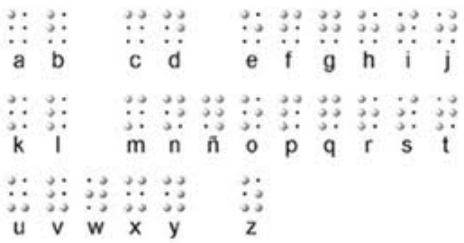

Figura 3.3: Alfabet braile

- **Mancances de visió**
  Una pèrdua de visió lleu, catarates cròniques, túnels, punts negres. Les mancances de visió comprenen qüalsevol disfunció que impedeixi veure correctament induïnt problemes de visió.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

- Pàgines web amb tamanys de font absoluts que no canvien fàcilment a l'ampliar o reduir la finestra
- Pàgines web o imatges de pàgines web que no disposen del contrast adequat, l'estil de les quals no pot ser sobreescrit per l'usuari amb estils propis
- Text mostrat com a imatges
- També algunes de les barreres mencionades en l'apartat anterior de ceguera, depenent del grau d'afectació de la mancança

Figura 3.4: Exemple de disminució visual

- **Daltonisme**
  El daltonisme és una mancança de percepció de certs colors.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Colors que són utilitzats com a únics enfatitzadors de text de la web
  - Text amb un contrast inadequat amb el color de fons o els patrons de colors que segueix la web
  - Navegadors que no permeten sobrecarregar els fulls d'estil proporcionats per l'autor de la web

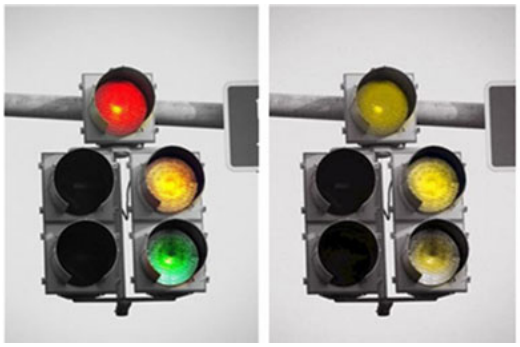

Figura 3.5: Exemple de daltonisme

3.3.2 Físiques

- **Discapacitats motores**
  Les discapacitats motores comprenen discapacitats com pèrdua d'extremitats, limitacions de moviment... [4]
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Opcions de resposta limitats en pàgines web
  - Navegadors i eines que no proporcionen alternatives al teclat per als events de ratolí
  - Formularis que no poden ser navegats en un ordre coherent

Figura 3.6: Arquitectura de l'accessibilitat a Java

3.3.3 Cognitives

Aquestes són les discapacitats relacionades amb la salut mental. A continuació s'enumeren les reconegudes actualment en les pautes del WAI:

- **Mancança de percepció visual o auditiva**
  Són discapacitats amb dificultat per a transcriure la informació rebuda pels òrgans visuals o auditius. Per exemple, els discapacitats amb dislèxia.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Pàgines web on la informació només es mostra d'una manera. Haurien d'haver varies maneres de mostrar aquesta informació

- **Mancança d'atenció**
  Els discapacitats amb manca d'atenció tenen dificultats per centrar-se en certa part important de la pàgina que es mostra.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Elements visuals o auditius que distreuen l'usuari i no poden ser desactivats d'una manera fàcil i intuïtiva
  - Organització inconsistent de la web

- **Mancança intel·lectual**
  Aquests són discapacitats amb una capacitat d'aprenentatge molt lenta. Inclús un cop l'aprenentatge esta assumit, tenen problemes per entendre conceptes complexes.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Ús innecessari d'un llenguatge complex a la web
  - Falta d'informació gràfica a les webs
  - Falta d'organització de l'informació a les webs

- **Mancança de memòria**
  Aquests són discapacitats que no recorden accions o decisions dutes a

terme recentment.
La barrera que aquest tipus de discapacitats poden trobar-se a la web es la falta d'una organització clara i consistent a la web.

- **Problemes de salud mental**
  Els discapacitats amb problemes mentals com l'esquizofrènia tenen problemes d'atenció i/o dificultat de visió (visió distorsionada) de la pàgina web per culpa de la medicació que es prenen.
  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Elements visuals o auditius que distreuen l'usuari i no poden ser desactivats d'una manera fàcil i intuïtiva
  - Pàgines web amb tamanys de font absolut que no pot ser canviat fàcilment de tamany

- **Problemes epilèptics**
  Els discapacitats amb problemes epilèptics són sensibles a els canvis constants de pantalla o senyals d'àudio a certa freqüència.
  La barrera que aquest tipus de discapacitats poden trobar-se a la web és, com s'ha dit, els canvis constants de pantalla per a vídeo o les senyals d'àudio a certa freqüència. Per tant s'han de evitar.

3.3.4 Auditives

[5]

- **Sordesa**
  La sordesa implica una disminució molt alta de l'oïda a ambdues orelles. El llenguatge que utilitzen els discapacitats sords és un llenguatge de símbols i a més a més és probable que els hi costi escriure o pronunciar paraules.

  Les barreres que aquest tipus de discapacitats poden trobar-se a la web són:

  - Falta de codificació de l'àudio en el llenguatge de símbols

  - Pàgines web amb abundància de text. Aquest hauria de ser substituït per imatges per resumir-lo
  - Text difícil d'entendre i poc clar
  - Pàgines web en les que es demana entrada d'àudio

- **Mancança de capacitat auditiva**
  Són discapacitats amb una reducció de l'oïda fins a un nivell moderat. La barrera que aquest tipus de discapacitats poden trobar-se a la web és també una de les anteriors: la falta de codificació de l'àudio en el llenguatge de símbols.

3.3.5 Comunicació

- **Discapacitats de pronunciació**
  Aquests són discapacitats els quals tenen problemes de pronuncia, ja sigui perquè no parlen prou clar o en un to de veu prou fort.
  La barrera que aquest tipus de discapacitats poden trobar-se a la web són pàgines en les quals es requereixi d'alguna entrada de àudio per a dur a terme alguna acció.

3.3.6 Discapacitats múltiples

Quan es dóna el cas que un mateix usuari agrupa diferents discapacitats sorgeixen noves necessitats, no contemplades si es tracten les discapacitats per separat.

Així quantes més discapacitats tingui l'usuari més difícil és fer el sistema accessible. Per exemple, un usuari cec i sord a la vegada necessitaria d'un sistema braille refrescable per a accedir a continguts multimèdia.
L'edat sol és un factor clau per l'aparició de diferents discapacitats en diferents graus.

3.4 Productes de tecnologia assistiva

L'accessibilitat a la informàtica és proporcionada per diversos productes, anomenats de tecnologia assistiva. Aquests productes es poden agrupar en dues grans famílies: els productes hardware i els productes software.

3.4.1 Productes hardware

Els productes hardware són dispositius que complementen l'ordinador amb les funcionalitats assistives que els seus usuaris requereixen. Com que aquests dispositius han de suplir les carències d'accessibilitat del sistema informàtic que s'està emprant solen ser dispositius especials d'entrada o bé de sortida. A continuació es llisten els dispositius de tecnologia assistiva més comuns.
Dispositius de tecnologia assistiva d'entrada:

- Teclats amb tecles més grans
- Teclats amb menys tecles
- Teclats amb configuracions de tecles diferents
- Micròfon (alternativa al teclat i/o ratolí)
- Eye tracker (alternativa al ratolí)
- Footmouse (alternativa al ratolí)
- Pantalla tàctil (alternativa al teclat i/o ratolí)

Dispositius de tecnologia assistiva de sortida:

- Dispositiu braile (alternativa a la pantalla)
- Auriculars / altaveus (alternativa a la pantalla)

3.4.2 Software

Hi ha funcionalitats que, o bé no es poden oferir a través d'uns dispositiu hardware, o bé un dispositiu hardware requereix de software especial per a funcionar correctament. Aquestes funcionalitats, que com ja s'ha dit poden ser complementàries d'un dispositiu hardware o no, també es poden dividir en els mateixos dos grups que els productes de tecnologia assistiva hardware.

Productes de tecnologia assistiva software d'entrada:

- Speech To Text ó reconeixement de veu (alternativa al teclat i/o ratolí, requereix d'un micròfon)

- Text predictiu
  Tecnologia que ajuda a l'usuari a escriure, tot intentant endevinar quina paraula es vol escriure. S'utilitza sobretot en dispositius mòbils i processadors de textos. (complementa el teclat).
- Teclat en pantalla (alternativa al teclat)
- Moure el ratolí amb el teclat (alternativa al ratolí)

Productes de tecnologia assistiva software de sortida:

- Lectors de pantalla via Text To Speech (alternativa a la pantalla)
- Magnificadors de pantalla / lupes
- Senyals visuals relacionades amb events del sistema (alternativa a l'alerta per altaveus)
- Modificació de l'interfície gràfica (augment del tamany de lletra de les icones, augment dels contrastos entre colors, canvi de colors per evitar els efectes del daltonisme i d'altres discapacitats visuals...)

Per tal que el software assistiu funcioni correctament i es pugui comunicar amb el hardware es fa necessària una capa de software intermediària, la responsabilitat de la qual recau al sistema operatiu. Una conseqüència directa d'això és que diferents sistemes operatius aporten diferents solucions a aquest problema. Aquesta capa de software s'encarrega de proporcionar tant al software assistiu com als drivers dels dispositius de tecnologia assistida una API amb la que treballar, així com software comú que ajudi al seu desenvolupament.

La capa de software que aporta el sistema operatiu rep el nom de proveïdor de serveis de tecnologia d'assistència.

Tot seguit es llista els proveïdors de serveis de tecnologia d'assitència més comuns:

- **Microsoft Active Accessibility (MSAA)**
  Per a Microsoft Windows, la primera versió sorgí amb Windows 95
- **Microsoft UI Automation**
  Per a Microsoft Windows (Native UI Automation) i .NET (Managed UI Automation), inclou MSAA 3.0

- **IAccessible2**
  Per Microsoft Windows. Creada per IBM com a alternativa a Microsoft UI Automation. És lliure.

- **MAC OS X Accessibility**
  Per sistemes Apple amb MAC OS X

- **AT-SPI**
  Dissenyada per a sistemes UNIX / Linux. La seva implementació a sistemes GNU/Linux recau sobre l'entorn d'escriptori, i més concretament sobre les llibreries gràfiques que utilitzen. AT-SPI està implementat a GTK+2 (gnome) i QT4 (KDE). Va ser desenvolupada per gnome.

Actualment, a més a més d'oferir proveïdors de tecnologia d'assistència, els sistemes operatius també inclouen programari bàsic d'accessibilitat. Entre el programari accessible més comú que podem trobar als diferents sistemes operatius tenim magnificadors de pantalla, teclats en pantalla, notificacions auditives del sistema...

Un exemple clar de l'ús d'aquestes tecnologies és el sistema operatiu per a dispositius mòbils Android.

Capítol 4

Java a l'actualitat

4.1 Introducció a Java

Java és un llenguatge de programació orientat a objectes amb quinze anys d'història. Va ser creat per Sun Microsystems (recentment absorbida per Oracle).

Java està pensat per a ser completament multiplataforma (gairebé tots els sistemes operatius actuals) i disposa de diferents versions per a diferents arquitectures (microelectrònica, ARM, x86, AMD64, etc). Tot això gràcies a la seva modularitat.

Java té aquestes característiques gràcies al concepte de la màquina virtual ja que els programes s'executen dins d'aquesta màquina, separant-los completament de l'arquitectura i el sistema operatiu. Hi ha diferents versions de màquines virtuals, una per a cada arquitectura/plataforma suportada.

Segons la recopilació de dades de diversos portals de codi, Java és el llenguatge més utilitzat actualment.

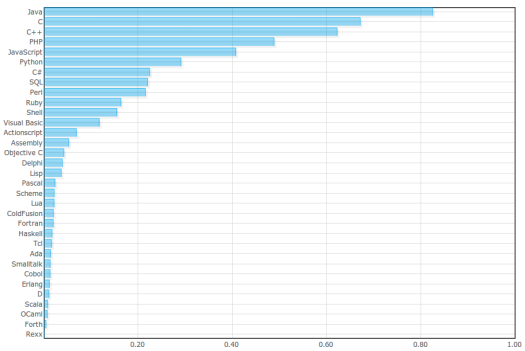

Figura 4.1: Comparació normalitzada de popularitat de diferents llenguatges de programació, segons www.langpop.com

4.2 Diferents dispositius

Així doncs hi ha moltes especificacions o plataformes Java, algunes són subconjunts d'unes primeres. Altres són totalment diferents i no comparteixen res més que certes classes abstractes.

Això és degut a la varietat de hardware existent en el que es pot executar un programa fet en Java. No fa falta mencionar les diferències entre un dispositiu mòbil, un ordinador de sobretaula o una nevera que realitza petites connexions a internet o controlar la temperatura. Apart d'aquests aspectes la plataforma Java ha de solucionar i abstreure problemes de hardware: des de tamanys de pantalles, capacitats gràfiques, capacitats de connexió del dispositiu, capacitats de processament, mecanismes d'E/S, i molts més aspectes.

A més a més no n'hi ha prou en les diferències de hardware, sinó que també ha de tenir en compte les diferències en quant a software i les funcionalitats que aquest software li ofereix. Per exemple, un sistema operatiu pot donar a Java certs privilegis d'accès a funcionalitats del 'driver' d'accès a hardware que en un altre sistema operatiu és impensable.

A tot això se li ha de sumar el fort creixement en l'actualitat de la computació distribuïda per Internet.

En el pròxim apartat s'enumera com la plataforma Java enfoca tots aquests requisits.

4.3 Heterogeneïtat de Java

Com hem dit anteriorment Java es troba disponible en més d'una versió i en diferents plataformes. Aquesta realitat, la d'un parc molt fragmentat, fa que desenvolupar aplicacions Java que funcionin a la majoria dels sistemes sigui un repte quant a usabilitat i accessibilitat ja que una solució comuna no sol ser la més eficient. El fet de tindre que enfrontar-se a sistemes diferents a l'hora de desenvolupar una aplicació sempre és un problema.

4.3.1 J2SE

J2SE[1] és la part de Java destinada als ordinadors de sobretaula. Actualment hi ha JVM's (Java Virtual Machine(s)) per a Windows, GNU/Linux i OSX, en diverses arquitectures. Aquesta versió incorpora un conjunt de classes força extens, i a més la comunitat l'extèn mitjançant llibreries dia a dia. Té moltes facilitats per a les estructures de dades, gràfics, connexió en xarxa, etc.

Tot això permet que sorgeixin aplicacions J2SE d'escriptori realment potents.

Encara que la tendència està en fer les aplicacions web, hi ha casos en els quals fer les aplicacions d'escriptori és l'única opció, degut a les limitacions del navegador i la connectivitat de la xarxa: latències, permisos, ubicació...

Han sorgit eines com GWT que permeten programar dins el paradigma de les aplicacions d'escriptori en J2SE de tota la vida però que, al ser compilades i empaquetades, són desplegades en un servidor d'aplicacions. Això ajuda a que programadors sense experiència en aplicacions J2EE puguin fer aplicacions basades en web utilitzant les tècniques que han fet anar sempre per aplicacions d'escriptori.

4.3.2 J2ME

J2ME[2] és una versió reduïda de J2ME. La seva màquina virtual suporta un subconjunt de les operacions que té la versió J2SE. Per exemple, no suporta operacions en punt flotant, en termes de programació, *float*. Això és degut a que els dispositius per als que esta encarada, tot i ser molt diversos en

[1] Java Standard Edition - http://es.wikipedia.org/wiki/Java_SE
[2] Java Mobile Edition - http://es.wikipedia.org/wiki/Java_Micro_Edition

quant a hardware, tenen un punt en comú: els seus processadors tenen poca capacitat de computació i força restriccions hardware per temes de consum d'energia.

Per pal·liar els problemes que suposa un rang tan extens de dispositius l'arquitectura de J2ME és molt modular i proporciona a la plataforma una gran flexibilitat i adaptació als diferents dispositius. Disposa de cinc nivells:

- **Pont amb el Sistema Operatiu**
  Ja que la plataforma no disposa d'accès directe a hardware.
- **Màquina Virtual**
  - **KVM**
    Kylobyte Virtual Machine és la màquina virtual per a dispositius de -molt- poca potència, com ara dispositius mòbils antics (per exemple de fabricació anterior al 2008).
  - **CVM**
    Compact Virtual Machine és la màquina virtual per a dispositus que tenen una capacitat de computació una mica més elevada i permissiva, com ara PDA's.
- **Configuració**
  Són les APIs que interactúen amb les diferents VM, oferint diferents funcionalitats que són comunes entre varis dispositius per la seva semblança en quant a hardware i capacitats.
  N'hi ha de dos tipus:
  - **CLDC**
    És el conjunt de llibreries que utilitzen els dispositius amb KVM. Al tenir limitacions hardware son funcionalitats pensades per a treballar en un dispositiu de poca capacitat computacional. Aquestes limitacions són per exemple dispositius amb una arquitectura de processador de no més de 32 bits, o dispositius que acostumen a funcionar sempre mitjançant la bateria i no estan connectats a la corrent. Requereix però un mínim de recursos per a poder funcionar: 190KB de memòria RAM mínim dedicats a la KVM.
  - **CDC**
    És el conjunt de llibreries que utilitzen els dispositius amb CVM. Aquestes permeten l'ús de funcionalitats més potents que CLDC

ja que els dispositius on hi és instalat tenen més capacitat de computació. Requereix com a mínim 2MB de memòria RAM dedicats a la CVM.

- **Perfils**
  Es podria dir que els perfils estan molt poc per sobre de la configuració ja que també són un conjunt d'APIs, però aquest cop, més restrictives i representen funcionalitats d'un subconjunt més concret que els dispositius de l'anterior punt. Degut al poc ús de CDC en el mercat, només cal mencionar que MIDP (Mobile Information Device) és el perfil més utilitzat.

- **Paquets opcionals o APIs propietàries**
  Són punts d'extensió dels perfils i són proporcionats per la plataforma. Generalment són API's funcionals aportades pel fabricant.

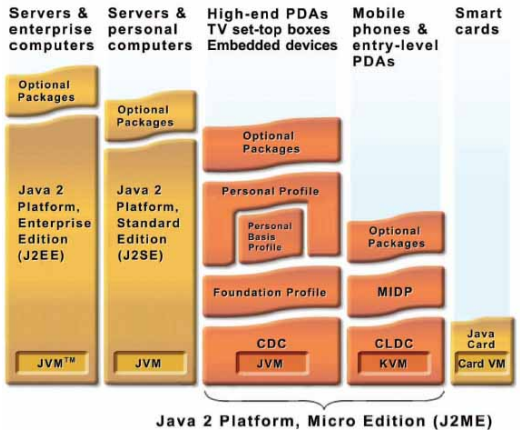

Figura 4.2: Desglosament de les capes de J2ME

De la mateixa manera que en una aplicació web, el seu projecte principal s'anomena EAR (Enterprise Application Project) en J2ME una aplicació mòbil s'anomena MIDlet (Mobile Information Device + l'equivalent a applet).

4.3.3 La preverificació en J2ME

[12] La verificació en Java és un procrès necessari per temes de seguretat.

El compilador de la VM corresonent converteix el programari fet en Java a instruccions bytecode que només entèn la VM. Per dir-ho d'alguna manera,

dins d'aquest bytecode estan les operacions binaries push, pop, multiplicacions, divisions, etc.

Un codi compilat podria tenir injectat bytecode perjudicial per la VM o ser compilat per un compilador 'no oficial' i d'aquí neix la verificació, que entre d'altres aspectes de seguretat, permet controlar els errors en temps d'execució i llençar-los en forma d'excepció, evitant que la VM o inclús el sistema operatiu on s'executa es comporti de forma no desitjada.

En la versió d'escriptori J2SE la verificació es fa en temps d'execució, moments abans de carregar el codi a executar el la JVM, sense afectar massa al rendiment del sistema.

En J2ME, degut un altre cop a la limitació de potència dels dispositius, la verificació s'ha dividit en dos parts i neix el concepte de la preverificació.

La preverificació consisteix en delegar gran part de la responsabilitat de la verificació a l'entorn de programació. Les eines que tenia la VM per verificar el codi, són passades a l'SDK de J2ME, permetent que just després de compilar el codi, es passi un segon constructor, que farà el checkeig de que el bytecode generat pel compilador compleix amb l'especificació de la JVM, versió Java i més accions.

4.3.4 Altres especificacions de Java

- **J2EE**
  Internet està de moda. Cada cop més les empreses basen el seu negoci en aplicacions Web, deixant de bada les aplicacions d'escriptori, costoses de distribuir i de mantenir.

  Els creadors de Java ho tenien clar, van fer la versió Enterprise[3], que conté tot el paquet J2SE i a més a més, incorpora una API per treballar en aplicacions Web.

  Introdueix el concepte de Servlet, que és un servei escoltant en un port en concret. En un mateix host pot haver-hi més d'una aplicació i per això sorgiren els servidors d'aplicacions per a Java. Gràcies a

[3]Java Enterprise Edition - http://es.wikipedia.org/wiki/J2EE

aquest middleware, pots disposar de múltiples aplicacions independents o inclús que es comuniquen entre elles, tot en una arquitectura de xarxa.

Una aplicació feta en J2EE s'anomena EAR (*Enterprise Application Project*[4]).

- **Java Card**
  La versió *Java Card*[5] de la plataforma Java permet l'execució de petites aplicacions java (*applets*) en tarjes intel·ligents generalment per a fer petites accions com pagaments, transaccions, i accions en general que necessitin un software segur i no gens pesat. Un exemple són les targetes GSM, alguns televisors, neveres, etc.

4.4 Diferents utilitats

El fet que Java estigui adreçat a un conjunt força important i heterogeni de dispositius fa que el conjunt d'aplicacions que hi ha implementades, així com el conjunt d'usuaris que les utilitzen siguin també heterogenis amb pols ben diferenciats.

Per exemple, podem trobar un usuari amb grans coneixements d'informàtica creant aplicacions per mitjà d'un entorn de desenvolupament que funciona amb Java, com és el cas d'Eclipse, al seu ordinador; d'altra banda podem observar (i cada dia més) un usuari accedint al seu correu electrònic[6], visitant una plana web (Opera Mini), o jugant una partida al seu joc preferit des d'el seu terminal mòbil, tot també gràcies a Java. Fins i tot podem trobar un usuari al seu sofà, amb el comandament a distància, navegant per un menú ric i interactiu d'una pel·lícula en format BluRay (J2ME forma part de l'estandard BluRay) al seu televisor.

Queda doncs demostrat que Java apareix a la vida de molta gent i de formes molt diverses. A més a més la forma i l'entorn en el que aquesta gent interactúa amb Java també pot ser radicalment diferent.

[4]EAR - http://en.wikipedia.org/wiki/EAR_(file_format)
[5]Java Card - http://es.wikipedia.org/wiki/Java_Card
[6]Gmail app - http://gmail.com

4.5 Java a la societat

Com s'ha dit abans Java és el més utilitzat, a continuació es fa un petit estudi de com esta la cosa actualment.

4.5.1 Empreses

Moltes de les empreses del sector IT (Information Technology o Tecnologia de la Informació) s'han passat a Java gràcies a la seva popularitat. És el cas d'Oracle, que recentment ha comprat Sun Microsystems, fundadora de Java.

Altres empreses que varen apostar fort per Java foren IBM creant l'IDE Eclipse, que ràpidament van oferir a la comunitat open source i varen treure una versió comercial basada en Eclipse, WebSphere.

La NASA també utilitza Java en les seves missions espacials i en la majoria dels seus aplicatius.

4.5.2 Comunitat

Gairebé tot el paquet Java és Open Source, això ha contribuït a la creació de fòrums, portals web, repositoris de codi font, etc dedicats a expandir Java.

Mitjançant el *Java Community Process*[7], JCP, les empreses poden sol·licitar definicions d'API per a futures versions de la plataforma.

Un dels principals exemples és l'aportació que ha fet durant els darrers anys la fundació Apache, aportant utilitats que ajuden i molt al desenvolupament d'aplicacions en Java. També es poden trobar milers de blogs parlant de Java per internet, cosa que demostra la seva popularitat.

4.5.3 Projectes

Els projectes més famosos fets en Java són els propis IDEs per treballar-hi: Eclipse i Netbeans.

[7]JCP - http://jcp.org/en/home/index

La plataforma Android és una clara aposta de com les grans empreses com Google utilitzen i confien en Java. Es tracta d'un sistema operatiu amb kernel Linux, filosofia open source i enfocat principalment a dispositius mòbils. La principal forma de crear aplicacions per a Android és mitjançant el seu SDK Java. Cal comentar, però, que el codi Java compilat per a Android no genera bytecode compatible amb altres màquines virtuals Java, ja que utilitza la seva pròpia implementació tant de la màquina virtual com de les llibreries Java.

D'aplicacions que també són molt utilitzades però ningú sap que estan fetes en Java són els TPV (Terminal Punt de Venda[8]) de multitud de caixers, tendes, restauració, etc..

[8]TPV - http://es.wikipedia.org/wiki/Terminal_punto_de_venta

Capítol 5

Accessibilitat en Java

5.1 Introducció

Fins ara s'ha posat al dia al lector informant-lo de l'actualitat en matèria de Java i d'accessibilitat.

En aquest capítol s'agrupen els conceptes posant exemples de quines són les eines que hi ha actualment per tractar l'accessibilitat en Java.

5.2 Java Access Bridge

El *Java Access Bridge* [1] és una llibreria de software que permet a la màquina virtual Java disposar de les tecnologies assistives que ofereixi el sistema operatiu. Aquesta llibreria per tant no és multiplataforma. Això porta un seguit d'inconvenients:

- Hetereogeneïtat en quant a funcionalitats i suport
- Suposa un pas extra a l'hora d'instal·lar software: no n'hi ha prou amb que sigui compatible amb el sistema, si no està instal·lada no podrem gaudir de les seves funcionalitats

Gràcies a aquesta llibreria programes com JAWS (Job Access With Speech[2]) poden obtindre la informació necessària per complir les seves tasques.

[1] Java Access Bridge - http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136191.html

[2] JAWS - http://www.freedomscientific.com/jaws-hq.asp

També ens permet utilitzar aplicacions Java amb teclats braille, etc (sempre que la aplicació sigui compatible).

5.3 JAAPI

La *Java Accessibility API*[3] és la peça de software que proporciona informació sobre la interfície d'usuari al Java Access Bridge. Aquesta informació s'obté o bé implementant aquesta API als nostres elements de la interfície d'usuari o bé utilitzant elements de la interfície d'usuari que ja la tingui implementada, com és el cas de qualsevol element de la llibreria gràfica Swing[4]. La JAAPI proporciona doncs a cada element de la interfície un context d'accessibilitat que pot ser llegit per les tecnologies assistives que així ho requereixin mitjançant el Java Access Bridge. Cal destacar que aquest context d'accessibilitat pot ser modificat per l'aplicació programàticament per tal d'alimentar a les tecnologies assistives amb informació més acurada. En la figura 5.1 es pot veure a nivell gràfic l'arquitectura de la JAAPI.

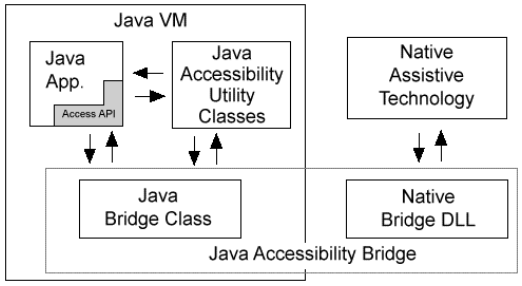

Figura 5.1: Els diferents mòduls que fan possible l'accessibilitat a Java

5.3.1 Guia d'accessibilitat IBM

Fa anys que IBM va desenvolupar (i encara manté i va actualitzant) una guia d'accessibilitat per a desenvolupadors Java. Aquest conjunt de pautes és molt complet i es pot trobar a la següent direcció web:
`http://www-03.ibm.com/able/guidelines/java/snsjavag.html`

Cal destacar la visió d'IBM quant a l'accessibilitat. Literalment de l'introducció de la guia podem llegir:

[3]JAAPI - http://download.oracle.com/javase/6/docs/technotes/guides/access/index.html
[4]JAAPI - http://en.wikipedia.org/wiki/Swing_(Java)

Els desenvolupadors d'aplicacions haurien de tindre en compte les necesitats especials dels usuaris discapacitats quan desenvolupen software en Java. Cobrir les necesitats d'aquests usuaris serà positiu per a tots els usuaris, incloent aquells que utilitzen dispositius comuns.

La guia ofereix un seguit de pautes generals necessàries per aconseguir una aplicació usable per a usuaris discapacitats. Aquestes pautes es centren sobretot en la forma en que l'usuari pot moure's entre les diferents accions que proporciona l'aplicació, com pot ser el suport d'una navegació per teclat senzilla i intuïtiva, utilitzar tècniques mnemòniques, dreceres de teclat, l'ús correcte de la presa de control, o *focus*, als elements de la interfície...

A més de les pautes genèriques la guia també proporciona una segona part, centrada en l'ús de Java Swing per millorar la accessibilitat de l'aplicació, tot explicant com és i com utilitzar la API d'accessibilitat implementada a Swing. Aquesta part també inclou un subapartat on explica de forma detallada com crear elements d'interfície propis accessibles, mitjançant l'ús de la API JAAPI.

Capítol 6

Estudi de les libreries d'Interfície d'Usuari a Java

6.1 Introducció

Durant el desenvolupament d'aquest projecte s'han implementat diversos adaptadors per algunes de les llibreries gràfiques més comunes de Java. La solució esta dissenyada de manera que es pugui extendre fàcilment per a poder suportar altres llibreries, mitjançant les interficies corresponents.

A continuació s'expliquen les llibreries que s'han implementat acompanyades cadascuna d'una o més imatges característiques de cada llibreria.

6.2 SWT

Standard Widget Toolkit[1] o SWT és la llibreria gràfica que ha adoptat el projecte Eclipse i és, conseqüentment, un dels pilars d'aquest framework.
El propòsit de SWT és oferir una API amb un bon rendiment un una aparença coherent amb el sistema operatiu sota el que s'està executant. Per assolir aquesta fita en comptes d'implementar tota una llibreria gràfica des de zero amb Java, aquesta llibreria fa un *wrapping* sobre les llibreries natives del sistema.

La seva última versió és la 3.6 (la mateixa que Eclipse).

[1] SWT - http://www.eclipse.org/swt/

Avantatges i inconvenients

El fet que SWT utilitzi les llibreries gràfiques que ofereix el sistema operatiu porta un seguit d'avantatges però també inconvenients respecte a una llibreria completament implementada en Java. El principal avantatge és que obviament l'usuari no pot distingir entre els elements del seu sistema dels d'una aplicació SWT. Així la coherència, cohesió i funcionament similars als que l'usuari espera. A més a més se suposa que les llibreries gràfiques natives estàn molt optimitzades i ofereixen menys sobrecàrrega al sistema respecte a qualsevol implementació Java. Gràcies a això SWT presumeix (tot i que sense una base massa sòlida de proves) ser la llibreria gràfica Java amb millor rendiment i menys impacte en els recursos del sistema.

Les desavantatges són diverses: es necessita una llibreria dependendent de la plataforma: el *wrapper*. Actualment la llibreria està disponible per a Motif[2], GTK+[3], GDI[4] (Windows) i Carbon[5] (Mac) principalment. Tot i que la llibreria està disponible per a la majoria de plataformes que hi ha actualment es fa necesari recalcar que es requerirà d'un desplegament diferent per a cadascuna d'elles, trencant així (tot i que només en part) un dels principals avantatges de Java: la multiplataforma.

A més a més hi ha una desavantatge que no és evident a primer cop d'ull: degut a que SWT delega el pes de la implementació a les llibreries natives del sistema existeix el risc que algunes de les seves funcionalitats no estiguin disponibles o que ho estiguin tan sols per a determinades plataformes, amb el que no es tindria forma de saber si els canvis realitzats a la interfície realment arriben a l'usuari del sistema o no, complicant la tasca d'anàlisi de resultats.

Figura 6.1: SWT sota Windows XP

[2]Motif - http://es.wikipedia.org/wiki/Motif
[3]GTK+ - http://www.gtk.org/
[4]GDI - http://msdn.microsoft.com/en-us/library/dd145203(VS.85).aspx
[5]Carbon - http://es.wikipedia.org/wiki/Carbon_(API)

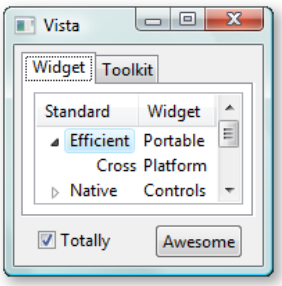

Figura 6.2: SWT sota Windows Vista

Figura 6.3: SWT sota Mac OS X

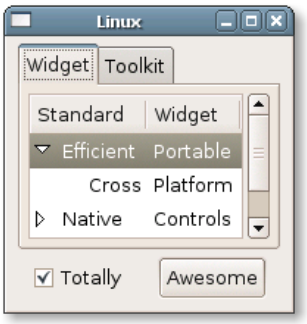

Figura 6.4: SWT sota GTK

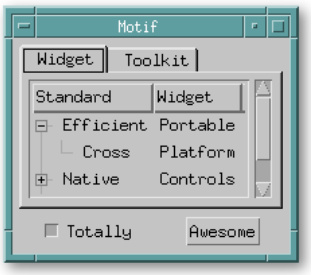

Figura 6.5: SWT sota Motif

Figura 6.6: SWT sota Photon

6.3 AWT

Abstract Window Toolkit[6] o AWT és una llibreria implementada íntegrament en Java que ofereix funcionalitats de finestra, *widgets*[7] (elements de la interfície gràfica) i gràfics (dibuixat de formes simples, com punts, línies, ...). Aquesta llibreria és universal a Java, això significa que allí on tinguem una màquina virtual J2SE podrem fer funcionar aquesta llibreria. A més a més, segons Wikipedia[REF correu], els perfils CDC requereixen que la màquina virtual J2ME inclogui AWT. Això és una molt bona notícia per la part de la solució mòbil del projecte.

AWT consta de dues parts ben diferenciades: una part que s'encarrega d'administrar les finestres, així com events i *layout managers*[8]. Aquesta és la part bàsica i tant Swing com Java2D es recolzen en aquesta part d'AWT

[6]AWT - http://es.wikipedia.org/wiki/Awt
[7]Widget - http://es.wikipedia.org/wiki/Widget
[8]Layout manager - http://en.wikipedia.org/wiki/Layout_manager

a les seves implementacions. L'altra part consisteix en un conjunt de widgets bàsics així com un canvas (un canvas és, explicat breument, un element al que s'accedeix i es dibuixa dins seu com si es tractés d'un *framebuffer*[9]), sobre el qual poden dibuixar altres llibreries.
Com que es distribueix juntament amb la màquina virtual, la versió oficial de AWT és la 6 (perquè l'última versió oficial estable de Java és la 6).

Avantatges i inconvenients
Al igual que SWT, AWT també es nodreix de la llibreria gràfica nativa per a generar els widgets. No obstant manca del principal punt feble de SWT (la pèrdua del compromís de multiplataforma que ofereix Java) al delegar aquesta funció, ja que hem de recordar que AWT s'inclou amb qualsevol màquina virtual J2SE des de la primera versió de Java, l'any 1995. De manera que a qualsevol màquina que puguem utilitzar J2SE tindrem aquesta llibreria disponible. El rendiment d'AWT és també molt alt. Un altre avantatge és la mimetització amb la resta de les aplicacions del sistema, exactament igual que amb el cas de SWT. D'altra banda, AWT no careix de desavantatges. La principal és que executant el mateix codi, el mateix programa oferirà diferents presentacions segons la plataforma sota la que s'estigui executant. A més a més probablement també tinguem el mateix problema que SWT, de no poder assegurar que els canvis sobre la interfície s'hagin aplicat en una plataforma concreta.

6.4 Swing

Swing, com AWT, forma part del JFC[10] (Java Foundation Classes), això vol dir que podem trobar-lo a qualsevol màquina virtual capaç de fer anar codi J2SE (Swing és part de JFC des de Java 1.2). No s'ha de prendre Swing com un reemplaçament de la llibreria AWT, sino com una extensió del mateix. De fet Swing utilitza els components bàsics d'AWT per manegar finestres i demés. També cal dir que els widgets de Swing i els d'AWT tenen una API força semblant. Swing consisteix en un conjunt de widgets que es dibuixen de forma nativa a Java (això és sense dependre de cap llibreria nativa del sistema) mitjançant Java2D[11], el qual també forma part de JFC.

Avantatges i inconvenients
Els avantatges són clars: es disposa de Swing en pràcticament qualsevol

[9]Framebuffer - http://en.wikipedia.org/wiki/Framebuffer
[10]JFC - http://java.sun.com/products/jfc/reference/faqs/index.html
[11]Java2D - http://java.sun.com/products/java-media/2D/index.jsp

màquina que pugui fer anar codi J2SE (perquè, tal com hem dit abans, forma part de JFC des de Java 1.2), qualsevol aplicació que utilitzi Swing a una interfície gràfica tindrà exactament la mateixa aparença a totes les plataformes (sempre i quan no s'indiqui a Swing que intenti emular el *look and feel* de la plataforma nativa).
Dins els inconvenients tenim que, al renderitzar-se íntegrament des de Java és una mica més pesat que les alternatives oferides per AWT i SWT, no obstant disposa d'un conjunt de widgets standard que sabem que es comportaran de la mateixa manera a qualsevol plataforma.

6.5 LCDUI

Limited Capability Device User Interface o LCDUI[12] és la llibreria gràfica oficial de la versió Mobile de la plataforma Java, J2ME. Actualment en la versió 2.5.2.

El concepte d'IU d'aquesta llibreria és força senzill: divideix la pantalla en dos parts, la part on es mostren els controls implementats pel programador, independents del dispositiu i una altra part on es mostren les accions que el programador desitgi, aquestes sí dependents del dispositiu on es mostren, que mapejen les tecles del dispositiu.

No permet més d'una pantalla activa en un moment donat, degut a les limitacions del harware on s'executen les aplicacions J2ME.

Avantatges i inconvenients
Els avantatges principals de LCDUI són disposar d'una API força lleugera per al dispositiu i que els seus controls són renderitzats per la màquina virtual i per tant funcionaran igual en qualsevol sistema operatiu. En cas de que la funcionalitat de l'interficie sigui senzilla és la millor opció a triar.

L'inconvenient és que s'ha quedat desfassada ja que els dispositius per als quals es va dissenyar han evolucionat moltíssim i les limitacions que els dissenyadors van posar-hi passen factura a les necessitats actuals. Tampoc soporta temes[13] i és el progrmador qui ha de portar la gestió de colors de cada pantalla i els seus controls.

[12]LCDUI - http://library.forum.nokia.com/index.jsp?topic=/Java_Developers_Library/GUID-0C6992B9-D044-48E2-8079-BD770E09010A.html
[13]Tema - http://es.wikipedia.org/wiki/Skin_(software)

A més l'API al ser tan lleugera és també molt limitada ja que per exemple no hi ha tractament d'events per als widgets, només per als Canvas. Això fa que si es necessita accedir a events com ara el generat al pitjar una tecla que no esta soportada per la llibreria s'ha de programar el widget mitjançant programació de baix nivell via la clase Canvas.

Figura 6.7: Exemple d'interfície feta en LCDUI

6.6 LWUIT

Light Weight User Interface Toolkit o LWUIT[14] és una alternativa de llibreria gràfica per J2ME. No és distribuida amb el Toolkit de J2ME per temes de compatibilitat però aquesta suporta i esta dissenyada per funcionar sobre MIDP 2.0 i CLDC 1.1 així com a les plataformes CDC. Per aquest motiu es pot descarregar i importar-se al projecte MIDLET sense cap complicació.

LWUIT proporciona una API potser no tan lleugera com la de LCDUI però molt més senzilla d'entendre per al programador, amb moltes més funcionalitats i basada en Swing. Gràcies a aquesta semblança LWUIT incorpora tot un sistema d'events per als widgets, permetent-los-hi accedir en qualsevol moment a qualsevol event generat per l'usuari com si d'una aplicació d'escriptori es tractés.

Gràcies a això ja no fa falta l'ús de la programació a nivell de pintat manual en comparació amb el LCDUI.

[14]LWUIT - http://en.wikipedia.org/wiki/Lightweight_User_Interface_Toolkit

La majoria d'aplicacions basades en J2ME que hi ha actualment al mercat utilitzen LWUIT per a l'interficie gràfica ja que les aplicacions fetes en un dispositiu en concret tenen el mateix emphlook'&'feel en un altre dispositiu sense necessitat de tocar el codi gràcies a l'us dels dels temes.

Avantatges i inconvenients
Avantatges:

- Tot el sistema d'events
- Molt familiar per al programador al ser molt similar amb altres llibreries gràfiques de la versió J2SE
- Permet l'ús de temes

Inconvenients:

- En uns dos anys o tres, ha quedat desplaçada degut a l'alta capacitat computacional del hardware actual, que porten sistemes operatius que suporten la versió J2SE. Per exemple els terminals amb Android[15] o Windows Mobile Edition[16]
- Que no vingui empaquetada amb la versió oficial de llibreries de J2ME

Figura 6.8: Exemple de LWUIT

[15]Android - http://www.android.com/
[16]Windows ME - http://en.wikipedia.org/wiki/Windows_Mobile

Figura 6.9: Exemple de LWUIT amb suport 3D

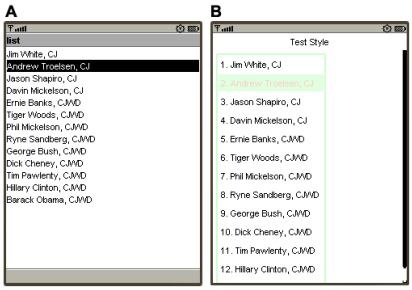

Figura 6.10: Comparació de la mateixa IU amb LCDUI (A) i LWUIT (B)

6.7 Estudi de les capacitats d'adaptació de les llibreries

Per tal d'estudiar les capacitats d'adaptació de les llibreries mencionades anteriorment s'han seleccionat els elements més habituals a una interfície d'usuari, i s'ha buscat per cada llibreria gràfica en concret un widget que les representi. Seguidament s'ha seleccionat un conjunt de canvis a fer a l'element, que són els que es pretén que el sistema acabi suportant.

En concret s'han buscat les següents característiques en els elements:

- Canvi de tamany
- Canvi del color dibuixat (foreground)

- Canvi del color de fons (background)
- Canvi de font (només per als elements que renderitzin text)

Es proposen doncs, dues taules per a cada element estudiat: La primera taula especificarà quin widget en concret s'ha estudiat per a cada llibreria. La segona taula indica quines de les característiques anteriorment llistades ofereix cada widget.

6.7.1 Text simple

Un element que mostra text simple. Pot respondre a accions de l'usuari, però no té perquè.

Widgets candidats:

SWT	Label
AWT	Label
Swing	JLabel
LCDUI	StringItem
LWUIT	Label

Suport a les llibreries:

	AWT	**SWT**	**Swing**	**LCDUI**	**LWUIT**
Tamany	SI	SI	SI	NO	SI
Foreground	SI	SI	SI	NO	SI
Background	SI	SI	SI	NO	SI
Font	SI	SI	SI	NO	SI

6.7.2 Botó

Un element que conté, com a mínim, un text i és capaç de rebre una acció concreta (la de ésser pitjat).

Widgets candidats:

SWT	Button
AWT	Button
Swing	JButton
LCDUI	(No suportat)
LWUIT	Button

Suport a les llibreries:

	AWT	SWT	Swing	LCDUI	LWUIT
Tamany	SI	SI	SI	-	SI
Foreground	SI	SI	SI	-	SI
Background	SI	SI	SI	-	SI
Font	SI	SI	SI	-	SI

6.7.3 Llista

Un element que presenta a la pantalla una llista d'elements de text. Els elements poden ser de tipus més complexos en algunes llibreries, però en aquesta prova només es demana que puguin mostrar text.

Widgets candidats:

SWT	List
AWT	List
Swing	JList
LCDUI	List
LWUIT	List

Suport a les llibreries:

	AWT	SWT	Swing	LCDUI	LWUIT
Tamany	SI	SI	SI	NO	SI
Foreground	SI	SI	SI	NO	SI
Background	SI	SI	SI	NO	SI
Font	SI	SI	SI	NO	SI

6.7.4 Arbre

Un element capaç de mostrar elements de text degudament ordenats en una estructura arborescent. Aquest elements poden ser més complexos o no, segons la llibreria gràfica.

Widgets candidats:

SWT	Tree (TreeItem)
AWT	List
Swing	JTree (TreeNode)
LCDUI	(No suportat)
LWUIT	(No suportat)

Suport a les llibreries:

	AWT	**SWT**	**Swing**	**LCDUI**	**LWUIT**
Tamany	SI	SI	SI	-	-
Foreground	SI	SI	SI	-	-
Background	SI	SI	SI	-	-
Font	SI	SI	SI	-	-

6.7.5 Entrada de texte

Un element que permet rebre text per part de l'usuari. Aquest element ha de poder rebre el focus del teclat del sistema.

Widgets candidats:

SWT	Text
AWT	TextField
Swing	JTextField
LCDUI	TextField
LWUIT	TextField

Suport a les llibreries:

	AWT	**SWT**	**Swing**	**LCDUI**	**LWUIT**
Tamany	SI	SI	SI	NO	SI
Foreground	SI	SI	SI	NO	SI
Background	SI	SI	SI	NO	SI
Font	SI	SI	SI	NO	SI

Capítol 7

Solució proposada

7.1 Introducció

Es proposa la solució que serà definida seguidament. En aquest capítol, però, només es mostra la solució proposada a nivell conceptual. Per veure els detalls de la implementació de la solució aquí proposada consulteu el següent capítol.

En el primer apartat s'explica la solució, conceptualment i a nivell global. Seguidament es desglossa la solució en els diferents blocs funcionals que la formen i s'exposa més detalladament quines són les funcionalitats que ha d'assumir cada bloc, com ja s'ha dit, sense entrar en detalls tècnics.

7.2 Proporcionar aplicacions que s'adapten dinàmicament

Es recorda que la solució ha de resoldre els punts llistats a continuació:

- L'usuari ha d'acabar obtenint una aplicació adaptada i preparada per oferir adaptacions dinàmiques a les seves necessitats
- L'adaptació ha de ser ràpida
- L'adaptació s'ha de calcular de manera transparent a l'usuari
- L'aplicació de l'adaptació ha de ser el més còmode possible per l'usuari

- L'adaptació ha de millorar en cada proposta
- L'adaptació ha de poder-se desfer
- L'adaptació ha de tindre en compte tant el dispositiu com l'usuari
- L'adaptació ha de ser lo suficientment bona per a poder ignorar la majoria de problemes de hardware possibles

Tenint en compte els requisits anteriors, es proposa un sistema que rebi aplicacions i sigui capaç de transformar les aplicacions que rep per fer-les adaptables i adaptatives. Aquesta transformació consistirà en la injecció al codi font de l'aplicació d'un bloc de codi especialment desenvolupat per a proporcionar aquesta capacitat. Aquest bloc injectat, que l'anomenarem component client, consta d'una part comuna per a qualsevol aplicació i plataforma i d'un mòdul que extèn aquest component per a que pugui funcionar amb l'aplicació, la plataforma i el dispositiu concrets als que està injectat.

Aquest bloc haurà de recollir dades tant de l'aplicació com de la interacció de l'usuari amb la interfície gràfica de l'aplicació. Aquestes dades seran útils a l'hora de proposar canvis a la interfície d'usuari. A més es proposa l'utilització de perfils d'usuari i dispositiu per proporcionar encara més dades. Quant més dades es puguin proporcionar al sistema millors propostes de canvis poden ser generades. Mitjançant totes les dades de les que disposi, el component servidor generarà propostes que seran oferides al component client el pròxim cop que arrenqui l'aplicació per mostrar-les finalment a l'usuari.

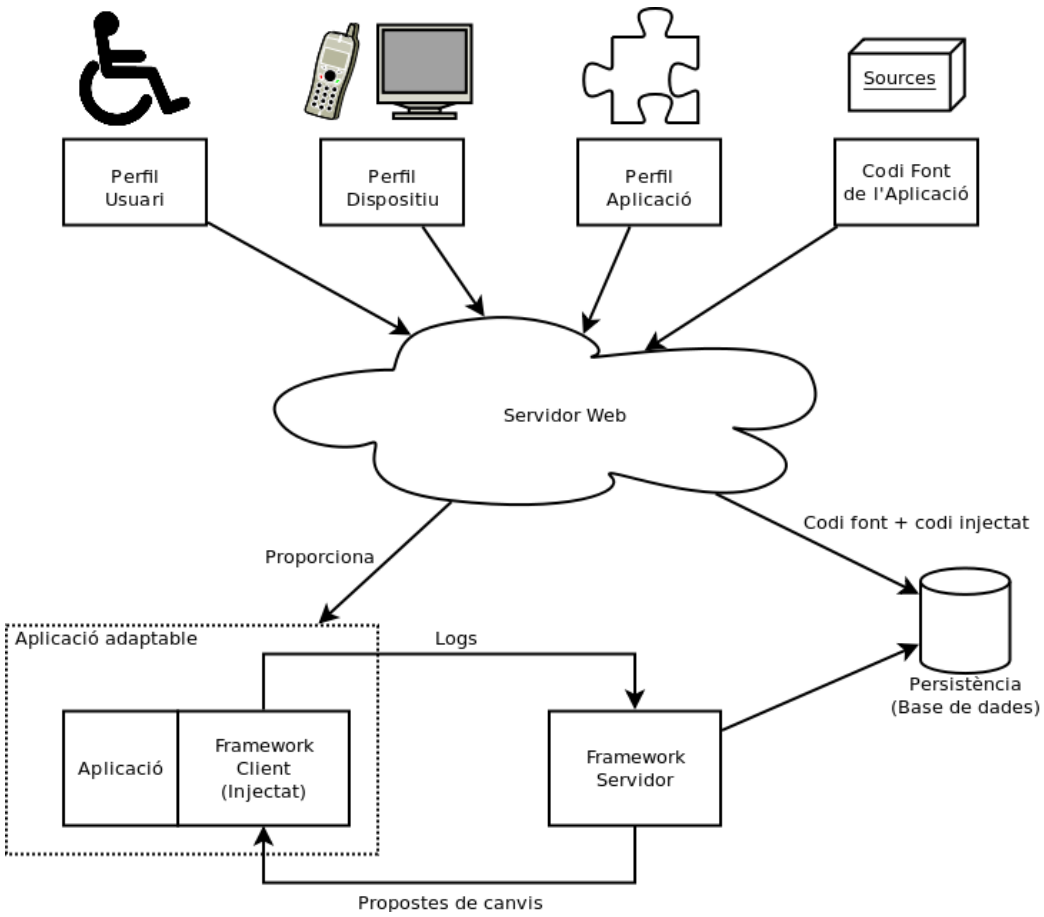

Figura 7.1: Diagrama general de la solució proposada

La solució proposada es divideix en tres blocs.

El primer bloc, la Web, és l'encarregat de l'administració de tota la solució, de recollir els diferents perfils, d'acceptar les aplicacions, afegir-hi el component client i d'oferir-les a l'usuari final.

El segon bloc, el component client, està íntimament relacionat amb el tercer, el component servidor. Ambdós treballen conjuntament per oferir a l'usuari final una aplicació, obtinguda prèviament des de la Web, adaptada dinàmicament.

El component client s'encarrega de capturar la forma en que l'usuari interactua amb la interfície d'usuari en forma de logs, enviar-los al component servidor, adaptar la interfície amb els canvis proposats pel component ser-

vidor i recollir feedback dels canvis realitzats. També s'encarrega d'oferir la funcionalitat de poder seleccionar, d'un historial de propostes de canvis, quina d'elles s'utilitzarà.

El component servidor es limita a tractar els logs enviats des del client i el feedback de l'usuari. Llavors a partir dels perfils d'usuari i dispositiu ofereix propostes de canvis a la interfície, els quals els envia al client i aquest s'ocupa d'aplicar-los si s'escau. La tasca de generar propostes de canvis a la interfície d'usuari recau a la peça clau del sistema: el motor d'inferència[1].

7.3 Aplicació Web

Per a poder dur a terme aquest punt de la solució, és a dir, per a recollir les dades dels perfils inicials i l'aplicació, s'ha optat per utilitzar un servei Web.

En la figura [FIGURA] es mostren tots els seus possibles casos d'ús que també es detallen a continuació:

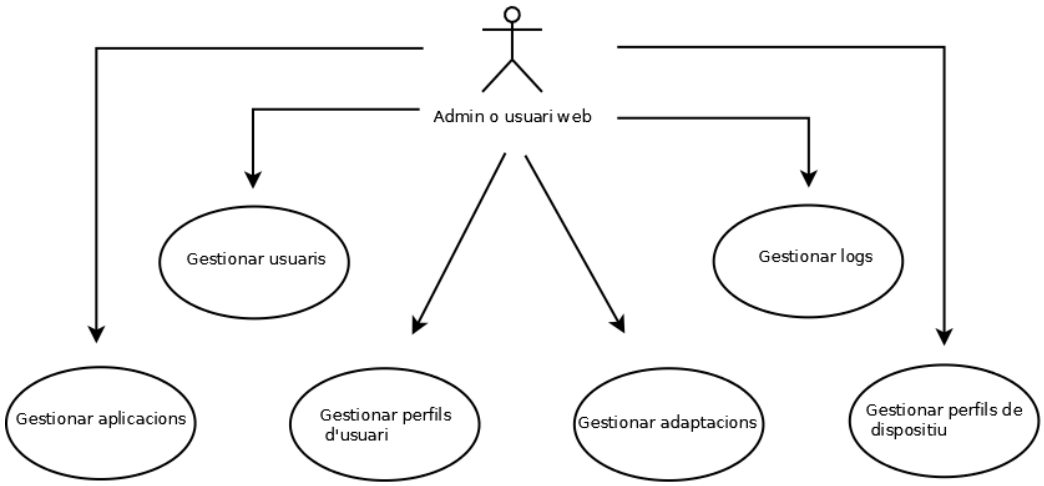

Figura 7.2: Diagrama de casos d'us del bloc Web

[1]Motor d'inferencia - S'explica en l'apartat 7.5.2

7.3.1 Funcionalitats

A continuació s'expliquen detalladament cada una de les funcionalitats que recauen en aquest bloc:

Gestionar usuaris

Tot i que l'aplicació Web encara no ho controla del tot, el model de dades està preparat perquè es pugui fer una gestió de l'usuari que esta connectat en sessió a l'aplicació.

Així doncs, qualsevol usuari que es registri haurà de proporcionar en un formulari si és administrador (per clarificar: administrador del discapacitat, no el conegut 'admin' o 'root' de qualsevol sistema) o be el propi discapacitat.

Aquest rol s'enmagatzema en persistència i s'utilitzarà en futures sessions per determinar quines accions es poden dur a terme per l'interfície de l'aplicació Web.

Gestionar perfils de dispositiu

Es disposa en persistència d'una llarga extensió de perfils de dispositiu. Aquests perfils contenen dades transparents a l'usuari necessàries per al correcte funcionament del sistema indicant-li quines capacitats funcionals té el dispositiu, ja sigui mòbil o d'escriptori.

Són fàcilment actualitzables ja que només cal actualitzar un fitxer XML de tamany reduït. També se'n poden crear de nous creats per algú que hi entengui editant aquest mateix fitxer.

A l'usuari se li mostrarà un llistat amb el nom de tots aquests dispositius disponibles en persistència i haurà de seleccionar-ne un. Un cop seleccionat és guardarà en persistència el perfil de dispositiu per a que l'utilitzin altres funcionalitats del sistema o be per a poder-lo editar posteriorment.

Gestionar perfils d'usuari

Des de la Web s'han de poder definir doncs els perfils d'usuari. En el capítol 3 es mencionaven una serie de discapacitats. D'aquestes s'ha considerat que eren aplicables pels objectius proposats pel sistema les discapacitats següents:

- **Fisiques**
  Aquestes engloben limitacions de qualsevol combinació de: mobilitat d'extremitats superiors, mans o bé dits. El sistema distingeix entre extremitat, mà i dits esquerres o bé extremitat, mà i dits drets.

- **Visuals**
  Aquestes engloben limitacions visuals com ara daltonisme, pèrdua de visió o ceguera total. El sistema distingeix entre ull esquerra o ull dret.

- **Cognitives**
  Aquestes engloben discapacitats com problemes de salud mental, epilèpsia o bé falta de memòria.

A part de tota aquesta informació, també es requereix de l'entrada al sistema de dades com el país de l'usuari i el sexe. Un cop les dades són guardades en persistència s'anomenen perfil d'usuari, que modelitza en base de dades l'usuari que utilitzarà les aplicacions a adaptar.

Aquest perfil serà utilitzat més endavant per altres funcions del sistema i també pot ser utilitzar per a aprofitar perfils, és a dir, poder-lo seleccionar d'un llistat en cas que l'usuari del qui es requereix la informació tingui un perfil idèntic d'un altre usuari prèviament insertat.

Cal mencionar que una ampliació de les discapacitats acceptades seria transparent a l'aplicació Web i al model de dades, limitant-se a actualitzar certs registres d'aquest últim.

Gestionar aplicacions

Per tal de poder oferir aplicacions adaptades, aquestes aplicacions han de ser prèviament definides al sistema. És necessari doncs omplir un formulari per tal que el sistema tingui totes les dades que necessita per poder adaptar l'aplicació.

Algunes de les dades més importants a entrar per l'usuari són:

- L'aplicació amb el codi font
- Llibreria gràfica utilitzada per l'aplicació
- Versió Java de l'aplicació

- Plataforma de l'aplicació
- Directori de dins el paquet de on es troba el codi font

Un cop introduïdes totes les dades del formulari el sistema ho guardarà en persistència com a perfil d'aplicació, que contindrà doncs l'aplicació en si i totes les dades importants per al sistema.

Injectar el component client a l'aplicació. Project Builder

Es té la necessitat doncs de disposar d'un instal·lable el primer cop que s'arrenca l'aplicació a adaptar.

La pròpia Web és capaç de compilar l'aplicació amb el component client injectat. Llavors emmagatzema l'aplicació ja preparada amb el nou codi en persistència, per tal de poder oferir-la a l'usuari mitjançant el gestor de descàrregues quan aquesta és requerida.

En aquest codi injectat s'hi posarà també un fitxer de propietats amb metadades per a que la comunicació entre client i servidor pugui dur-se a terme així com l'identificació de l'adaptació el primer cop que es connecta.

Gestionar regles del motor d'inferència

El Motor d'Inferència necessita estar alimentat de regles. Aquestes regles han de ser introduïdes al sistema per un administrador des de l'aplicació Web. Aquestes seran guardades en persistència.

Aquestes regles han de poder millorar-se mentre els usuaris van utilitzant l'aplicació ja que quant més l'utilitzin més informació tindrà tant el sistema com els administradors per tal de pensar nous canvis a fer.

Així doncs en qualsevol moment les regles prèviament definides poden ser recuperades i millorades mitjançant la selecció d'una d'elles en un llistat.

Gestionar logs d'usuari

Aquesta funcionalitat és bàsica pels administradors. Per tal de definir les regles del motor d'inferència que determinaran canvis els administradors necessiten consultar els logs que ha generat l'ús de les aplicacions, per poder així estudiar-los i decidir quins canvis poden ser propicis i en quines situacions.

A més l'estudi dels logs pot ajudar a identificar problemes d'usabilitat d'una aplicació, descobrir problemes que sofreix un usuari concret, etc, ja que els logs donen accés a informació tant de l'interfície d'usuari de l'aplicació com de la forma en que els usuaris interactúen amb ella.

Així doncs la monitorització de logs és una peça clau a tres bandes, ja que proporciona informació vital per extreure conclusions quant a la usabilitat de les aplicacions, la eficàcia dels canvis realitzats a la interfície i els usuaris que les utilitzen.

Més endavant s'explicarà el concepte de log d'una sessió de l'usuari en una aplicació. De moment n'hi ha prou en pensar que aquesta sessió esta modelitzada en persistència i que d'alguna manera el servidor ho enmagatzema. Un cop aquesta informació és al servidor des de l'aplicació Web es permet filtrar-los eficientment.

Algunes de les opcions de filtratge disponibles des del portal Web són:

- Usuari del log
- Adaptació
- Tamany, color o propietats
- Events produits
- Temps entre pantalles

Totes les acotacions permeten l'ús dels operadors més importants de la sintaxi SQL com `>,<, !=, between, like` i més.

Un cop filtrats es poden visualitzar per pantalla o bé es poden descarregar en format .txt per un estudi posterior.

Gestionar adaptacions

Pel model de dades una adaptació és un concepte que vincula una aplicació en concret, amb un usuari en concret, amb un dispositiu en concret i amb un conjunt de regles concret que proporcionaran propostes de canvi a la interfície d'usuari. Llavors un cop introduides totes les dades d'aquests perfils, aquests s'engloben en persistència en forma d'una adaptació única i identificativa. Aquest identificador d'adaptació serà l'utilitzat pel sistema per totes

les funcionalitats.

Un usuari de la Web només podrà obtindre l'aplicació amb el codi injectat si s'han introduït al sistema tots els perfils necessaris i per tant, existeix una adaptació.

Els administradors han de poder editar i esborrar adaptacions previament introduïdes al sistema per tal que els usuaris puguin gaudir de la resta de funcionalitats que que proporciona l'aplicació Web.

Descarregar aplicacions adaptades

Els usuaris tindran accés a una llista amb totes les aplicacions que tenen una adaptació pensada per ells en concret. El sistema facilita la descàrrega de l'aplicació adaptada mitjançant aquesta funcionalitat.

Aquesta funcionalitat esta disponible des de que s'ha completat el registre de l'aplicació com en les successives adaptacions que s'hi van produint.

Funcionalitats menors

Es consideren funcionalitats menors aquelles que no aporten cap solució als requisits del problema, però que enriqueixen l'experiència de la utilització de la Web.

- **Publicació / Presentació de notícies**
  El sistema proporciona les eines necessàries per crear / publicar notícies, en una estructura semblant a la d'un weblog.

- **Accés a la documentació del sistema**
  Aquesta funcionalitat resulta molt útil especialment per als administradors.

  Donat que la solució que es proposa és tan tècnica com conceptualment complexa s'ofereix a l'usuari de la Web múltiples fonts de documentació que poden ajudar-li a millorar l'experiència amb el sistema.
  Aquesta documentació es fa especialment útil per utilitzar la funcionalitat de la definició de regles pel motor d'inferència, ja que els administradors no tenen perquè conèixer els funcionaments interns del sistema, ni del motor d'inferència que implementi la solució, ni de com el motor

d'inferència té accés a les dades requerides per les regles.

Així amb aquesta documentació s'estableix un pont entre l'administrador i les funcionalitats que ha de utilitzar.

També hi és disponible aquest document.

- **Paginació**
  Tots els llistats són paginables per índex per a poder-los fer més usables per l'usuari, ja que redueix el temps d'espera i la navegació

7.3.2 Disseny de la Web

S'ha optat per un diseny minimalista en quant a colors i figures. Tot suavitzat amb contorns arrodonits.

La disposició de la informació en pantalla es divideix en tres blocs:

- **Capçalera**
  Conté el banner i el menú superior. En el menú superior es troben ubicades accions que poden utilitzar usuaris no registrats com ara notícies, descàrregues i un *quant a*.

- **Menú esquerra**
  Varia segons si l'usuari s'ha autenticat al sistema o no. Inicialment s'hi mostra el formulari d'autenticació, es dóna la opció de registrar-se com a nou usuari del sistema. Un cop autenticat l'usuari, el menú mostra les accions d'adaptacions, perfils d'usuari, perfils de dispositiu i logs.

- **Cos**
  És on s'hi mostren els resultats de les accions tant del menú esquerra com el menú superior.

A continuació és mostren vàries captures de les funcionalitats prèviament definides quedant així constància tant del disseny com de l'aplicació de cada cas d'ús.

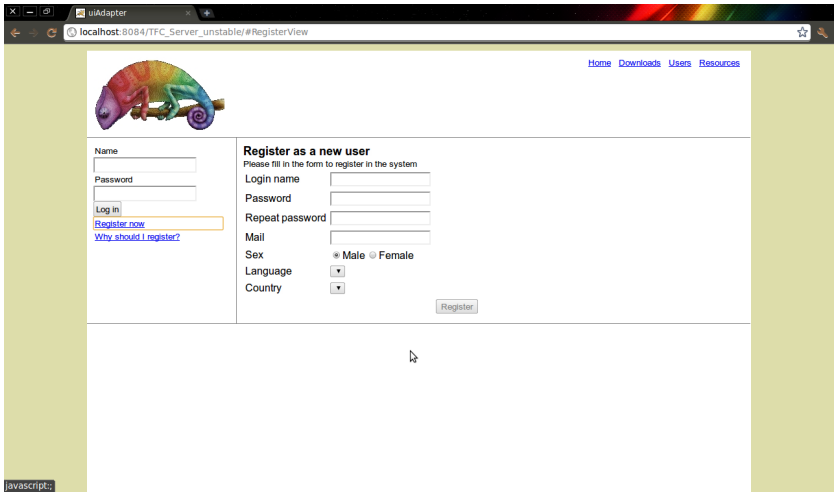

Figura 7.3: Gestió d'usuaris

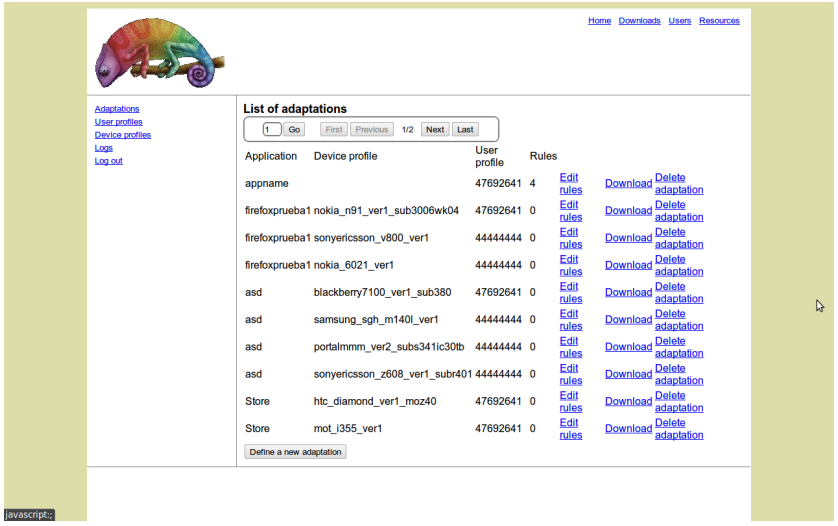

Figura 7.4: Gestió d'adaptacions. Llistat

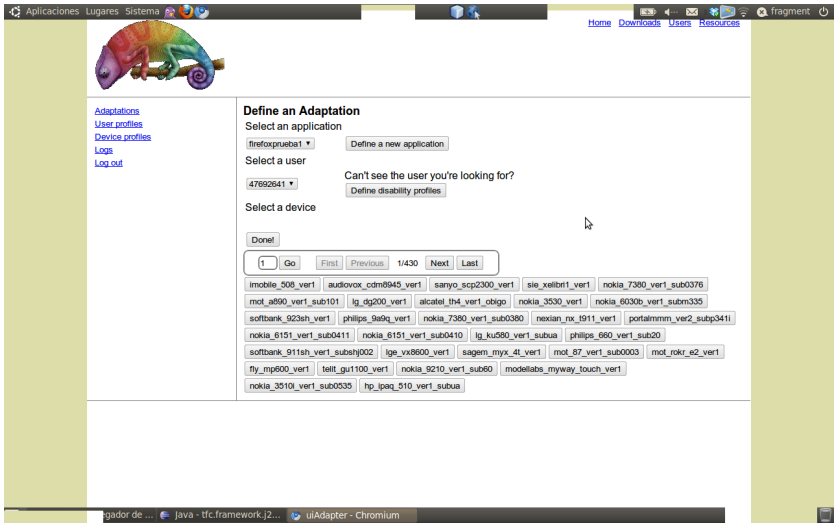

Figura 7.5: Gestió d'adaptacions. Definició.

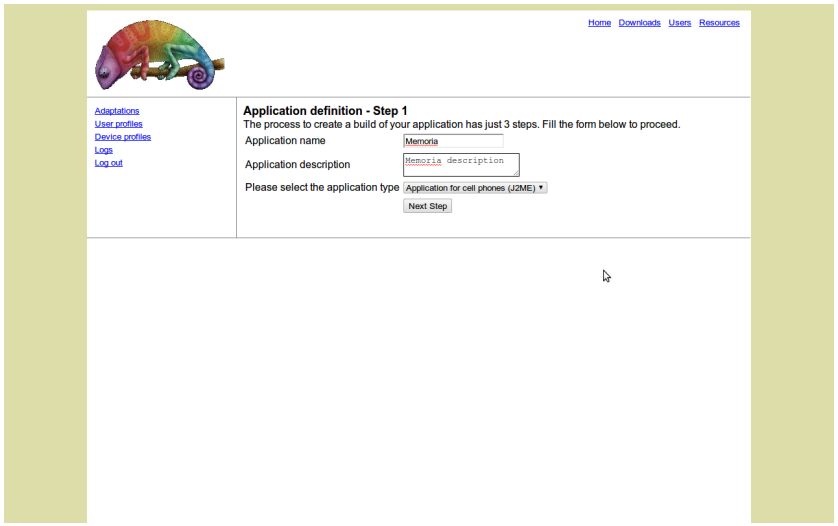

Figura 7.6: Gestió d'aplicacions

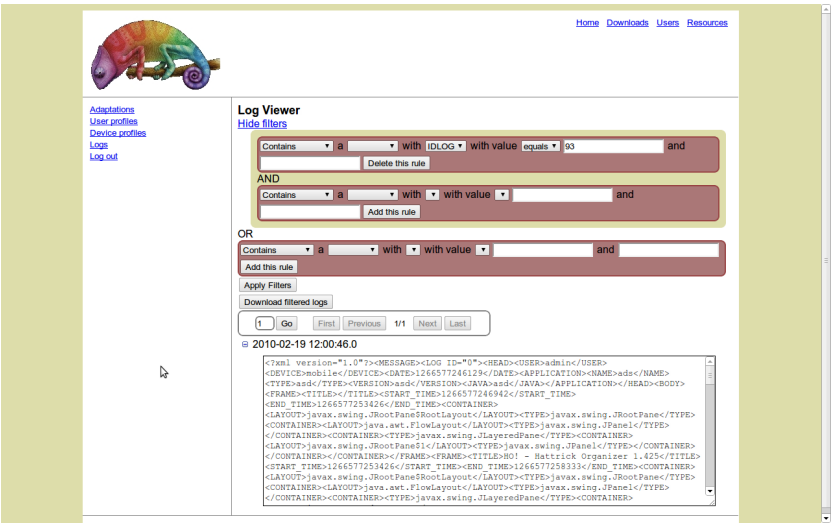

Figura 7.7: Gestió de logs

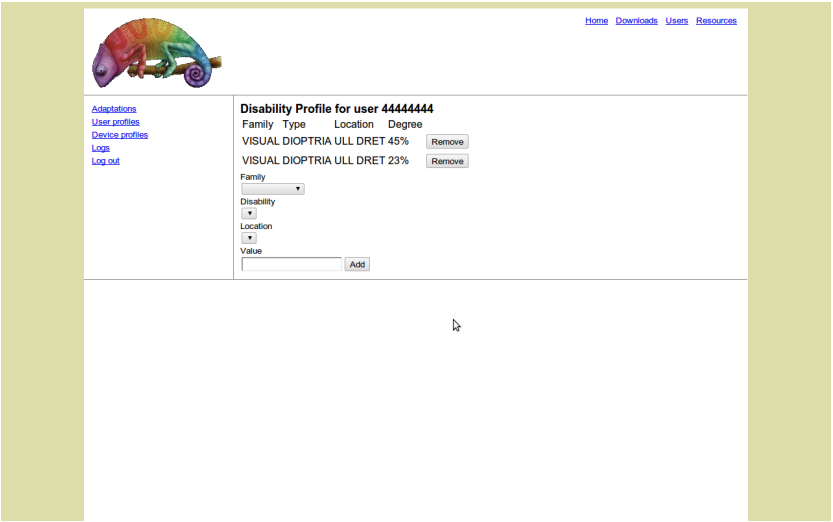

Figura 7.8: Gestió de perfil d'usuari. Llistat

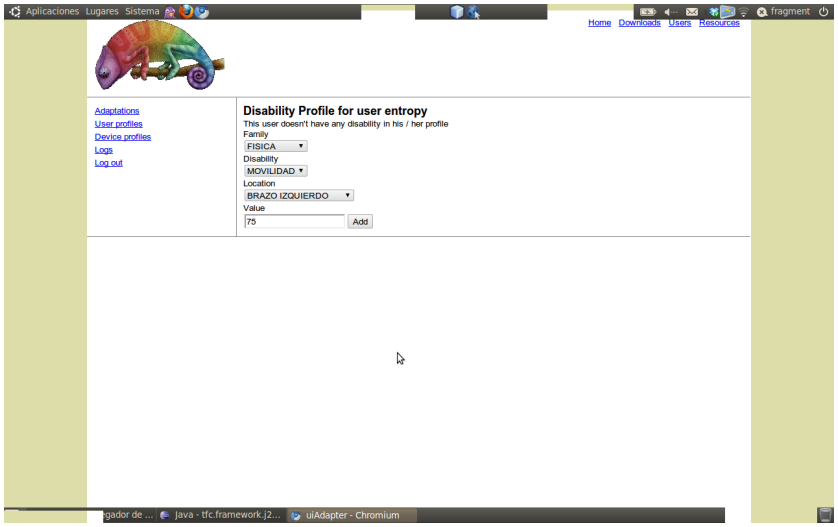

Figura 7.9: Gestió de perfil d'usuari. Creació

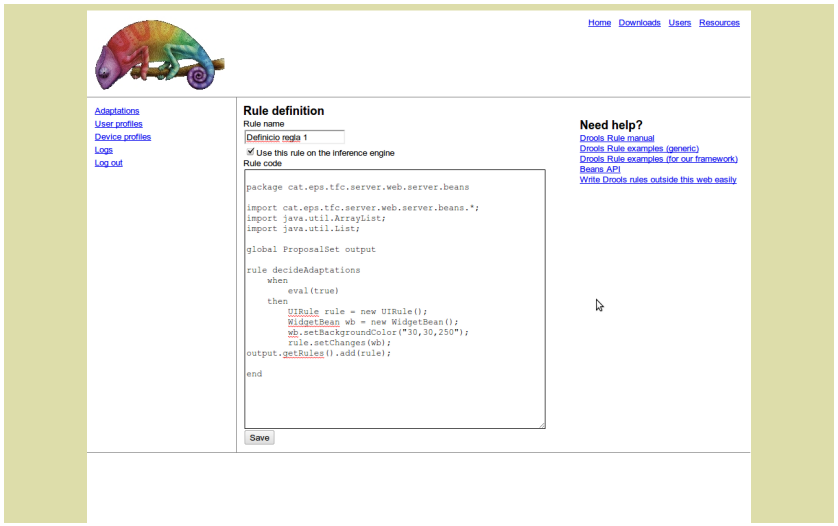

Figura 7.10: Definició de regles

7.4 Component client

El component client és la part de la solució que esta desplegada en el dispositiu de l'usuari. S'encarrega de monitoritzar les accions d'usuari i aspectes de l'aplicació per nodrir el motor d'inferència.

Així mateix també adapta la interfície d'usuari de l'aplicació amb els canvis proposats pel component servidor sempre i quan l'usuari els accepti a l'arrancar l'aplicació.

Opcionalment recull *feedback* dels canvis aplicats a la interfície d'usuari.

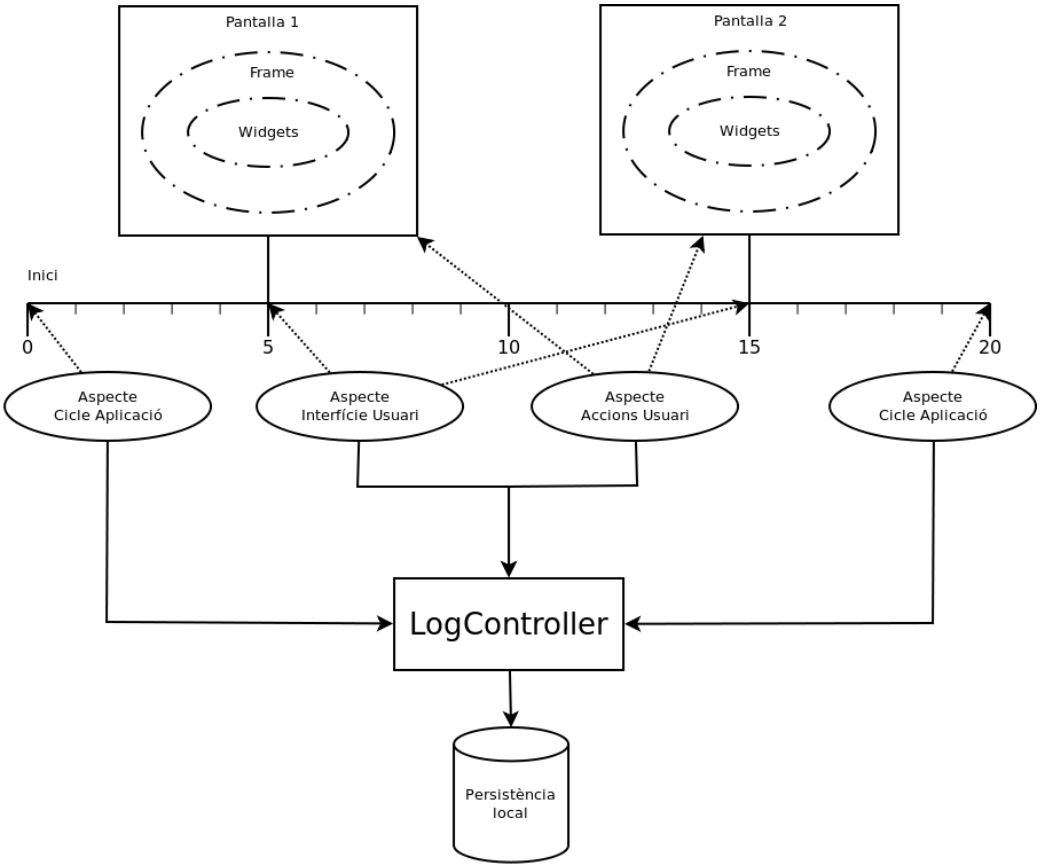

Figura 7.11: Recull de logs

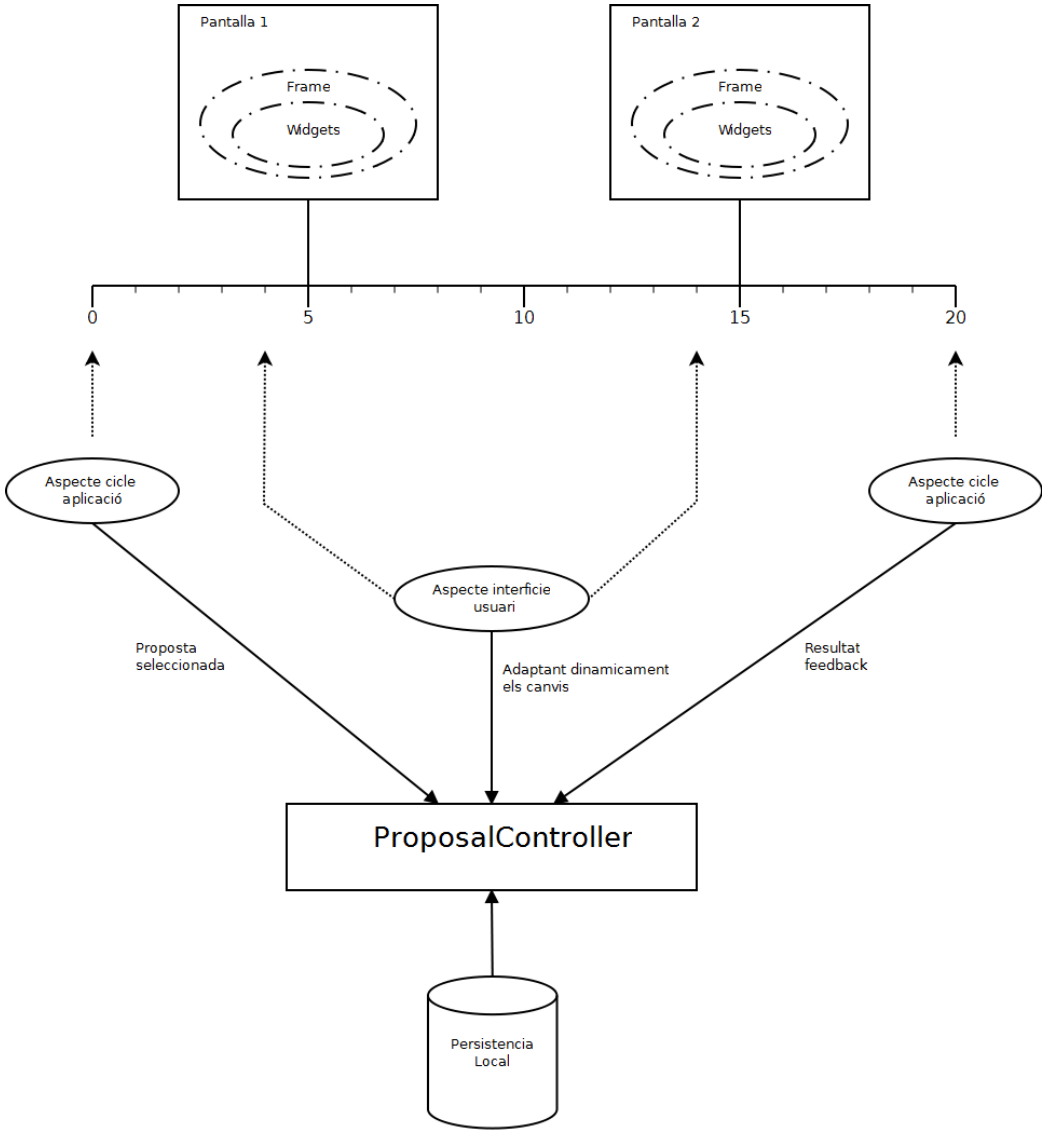

Figura 7.12: Adaptació de la interfície d'usuari

7.4.1 Funcionalitats

A continuació s'expliquen detalladament les funcionalitats que recauen a aquest bloc:

Monitoritzar el cicle d'aplicació

Quan l'usuari arrenca l'aplicació el component client intercepta la càrrega i abans de que es mostri la primera pantalla, comprova en la persistència del dispositiu si existeixen adaptacions prèvies. En cas de que n'hi hagi, es crea una pantalla.

El component client intercepta i logueja els mètodes de creació de l'aplicació i el de destrucció. A més també es pot enregistrar quan l'aplicació es pausa (en el cas d'aplicacions per a mòbils) o quan es minimitza / entra en segon pla (en cas d'aplicacions d'escriptori). Aquesta informació és indispensable per tal de que altres funcionalitats que s'explicaran posteriorment es puguin dur a terme.

A més també pot ser útil per a optimitzar algunes regles, ja que pot ser que un temps d'espera que un administrador consideri excessiu per realitzar una tasca sigui el resultat d'una trucada entrant, per exemple.

Monitoritzar l'interfície d'usuari

El component client té accés a tota la interfície d'usuari de l'aplicació i codifica la seva estructura en el log que es va omplint durant la sessió. Això implica que es monitoritzin totes les finestres , tots els components, totes les fonts, totes les imatges, etc.

Segons quina llibreria gràfica utilitza l'aplicació, el component client disposarà d'un mòdul específic per aquesta per tal de poder dur a terme aquesta codificació correctament. El sistema ha modelitzat l'estructura del log tal i com s'especifica en el protocol de comunicació explicat posteriorment, tot i que és ampliable. Per tant cada llibreria gràfica ha de disposar d'una implementació concreta d'aquest mòdul i l'estructura de les dades que recull han d'acabar essent la mateixa per a qualsevol llibreria gràfica.

D'aquesta manera en el servidor es disposarà d'una única representació de logs d'interficie d'usuari independent de la llibreria gràfica, facilitant així els seus mecanismes, com el motor d'inferència o la monitorització i descàrrega de logs, i brindant-los una visió abstracta i genèrica de log per a qualsevol llibreria gràfica.

Monitoritzar les accions d'usuari

Cada interacció de l'usuari amb l'aplicació queda monitoritzada en forma de log. Aquest recull de dades consisteix en calcular els intervals de temps entre acció i acció, com podria ser entre event i event. També es recullen els events relacionats amb la llibreria gràfica que s'utilitza. Les accions d'usuari també estan vinculades quasi totalment a la llibreria gràfica que empra l'aplicació.

De la mateixa manera que el recull de dades de l'interficie d'usuari, les accions d'usuari es registren en un format completament abstracte respecte a la llibreria gràfica emprada facilitant el seu tractament al servidor.

Les accions d'usuari són les dades principals de que es disposaran a l'hora d'escriure regles pel motor d'inferència. Així és important intentar recollir quantes més dades d'aquest tipus millor.

Seleccionar recull de canvis a aplicar

Conforme el component servidor va proposant reculls de canvis al component client, l'usuari ha de poder escollir quin recull de canvis vol aplicar a l'aplicació. S'ha decidit que el moment idoni per a escollir-ho sigui l'inici de l'aplicació, ja que així els canvis són els mateixos durant tota la sessió, facilitant la coherència per l'usuari en tot moment.

La selecció de recull de canvis a aplicar es proposa en forma de llista ordenada, essent més prioritaris els que l'usuari té en millor consideració, és a dir, ordenats o bé per feedback o bé pel nombre de vegades que l'usuari l'ha seleccionat. També fòra bo, però, que l'última proposta de canvi tingui prioritat sobre qualsevol altra.

Seleccionar feedback dels canvis aplicats

Opcionalment el sistema suporta la puntuació de la satisfacció de l'usuari respecte a un recull de canvis concret. Per tal de recollir el major nombre possible de puntuacions s'ha decidit que en comptes d'incorporar-ho a la Web, sigui el component client qui demani el feedback a l'usuari, integrant-se així en el cicle d'ús de l'aplicació.

S'ha decidit que el millor moment per demanar aquest feedback sigui quan l'aplicació es tanqui, ja que llavors tindrà probablement la opinió més objectiva possible respecte dels canvis que tot just ha disfrutat. Si no s'havia aplicat cap canvi l'usuari també té la posibilitat de puntuar l'ús de l'aplicació

sense haver sigut alterada, ja que aquesta nota pot servir com a referència amb la que comparar les posteriors puntuacions dels diferents canvis que es vagin aplicant.

Adaptar la interfície d'usuari

Les principals funcionalitats del component client són dos: s'ha vist que la primera era la monitorització de varis aspectes necessaris tant de l'aplicació com d'accions d'usuari. La segona és sens dubte l'aplicació dels canvis a la interfície d'usuari. Per tal d'aplicar els canvis el component client necessita novament d'un mòdul concret per poder interactuar amb la llibreria gràfica que utilitzi l'aplicació.

Els canvis s'apliquen en quant l'usuari escull un dels reculls de propostes de canvis. Si l'usuari no en selecciona cap o el component client encara no en disposa de cap aquesta funcionalitat es desactiva fins el proper cop que s'iniciï l'aplicació.

Enviar logs al component servidor

Els logs generats pel mòdul de monitorització han de ser enviats al servidor tard o d'hora per a que el component servidor disposi de quanta més informació millor per a proposar canvis. El moment en que s'envien els logs es deixa en mans de la implementació, ja que segons la plataforma, el dispositiu i el moment d'ús el moment més favorable per enviar dades pot variar enormement.

De totes formes els logs han de ser emmagatzemats en una persistència local (és a dir, al dispositiu on s'han capturat) fins que no es pot garantir que aquests han arribat correctament al component servidor. Un cop es puguin enviar al servidor, el component client accedirà a aquesta persistència i enviarà els logs al servidor.

Rebre propostes de canvis del component servidor

En un moment o altre, el component client ha de obtindre dades del component servidor, especificant nous reculls de propostes de canvis a aplicar.
Com ja s'ha explicat al punt anterior, el moment de connexió amb el component servidor es deixa en mans de la implementació. Per exemple, en l'implementació per a dispositius mòbils s'ha optat per a que sigui a l'inici de l'aplicació, mostrant una pantalla de selecció de canvis. Aquesta pantalla mostra un petit historial de les últimes propostes puntuades pel feedback de

l'usuari, deixant sempre l'opció d'escollir l'aplicació original i mostrant com a segona opció la nova proposta en cas d'existir.

7.5 Component servidor

El component servidor és el bloc que mitjançant la informació proporcionada pel component client i els perfils guardats en persistència, recollits per la Web, genera propostes d'adaptació per un usuari, dispositiu i aplicació en concrets.

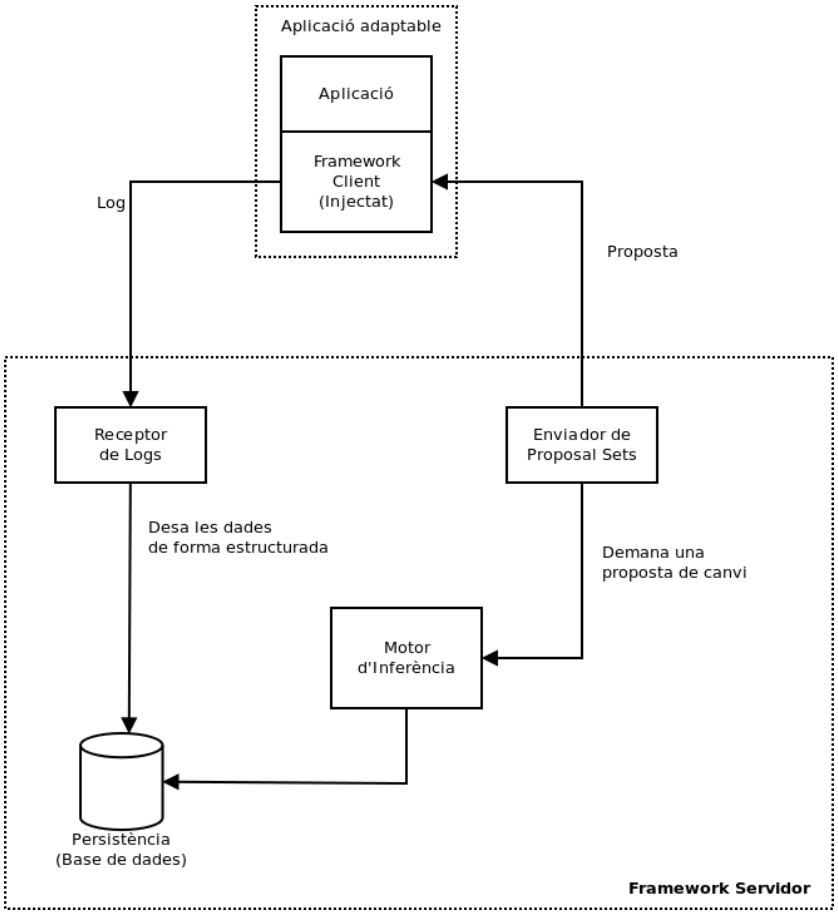

Figura 7.13: Diagrama del component servidor

7.5.1 Funcionalitats

A continuació s'expliquen detalladament les funcionalitats que recauen a aquest bloc:

Rebre logs del component client

Aquest bloc ha de recollir els logs preparats pel component client i emmagatzemar-los persistentment. A més ha de notificar d'alguna manera al component client que ha rebut correctament els logs per a que aquest pugui esborrar-los de la seva persistència local. Cal remarcar que, així com el component client pot permetre's el luxe d'emmagatzemar els logs en format cru, el component servidor ha d'emmagatzemar-los de forma estructurada ja que posteriorment hauran de ser consultats tant pel mateix component servidor, a l'hora de generar propostes de canvi, com per la Web, per tal de poder, per exemple, poder-los filtrar i mostrar-los a un administrador.

Generar propostes de canvis a la interfície d'usuari

Per tal de poder generar propostes de canvi a la interfície d'usuari, el component client fa ús d'un component software anomentat *Motor d'Inferència*. Aquest component, que s'explica amb més detall seguidament, es pot entendre com una caixa negra on, a partir d'unes dades d'entrada es generen unes dades de sortida, tenint en compte unes regles. Aquestes regles són les que ha d'introduir l'administrador per cada adaptació, via Web.

Quant a les dades d'entrada que s'han de proporcionar al Motor d'inferència per obtenir una proposta de canvis a la interfície d'usuari per a una adaptació concreta tenim les següents:

- Regles associades a l'adaptació
- Logs associats a l'adaptació (que inclouen tant la definició de la interfície d'usuari com les accions que ha realitzat l'usuari sobre la mateixa)
- Perfil d'usuari
- Perfil de dispositiu
- Canvis proposats anteriorment, amb el seu feedback corresponent

Enviar propostes de canvi al component client

Les respostes que s'han generat han de ser enviades al component client per tal que l'usuari es pugui beneficiar d'elles, així com jutjar-les.

7.5.2 El Motor d'Inferència

Al camp de l'intel·ligència artificial, un *Sistema Expert*[2] és un component software que intenta deduir conclusions a partir d'una base de coneixement. Una base de coneixement no és més que un conjunt de dades degudament codificat i introduït en un sistema informàtic. Un motor d'inferència és un sistema expert que, per tal de deduir conclusions, ho fa a través d'un conjunt de regles que han de ser introduïdes al sistema.

Les regles són expressions en llenguatge formal que són activades per elements concrets de la base de coneixement (especificats a la mateixa regla) i provoquen l'activació d'altres regles així com la definició (o alteració) de conclusions que retornarà el motor d'inferència. Les regles es podrien definir d'una forma més vulgar com mètodes de codi que no poden ser cridats, sinó que el sistema executa quan existeixen uns elements concrets amb un estat concret (especificat a la pròpia regla) a la base de coneixement.

Cal destacar doncs que les regles s'executaran tants cops com elements amb l'estat adequat existeixin a la base de dades, però també s'executaran quan una altra regla modifiqui un element portant-lo a un estat que coincideixi amb el definit a l'activació de la regla.

[2]Sistema Expert - http://es.wikipedia.org/wiki/Sistema_experto

Capítol 8

Tecnologies utilitzades i detalls de la implementació

8.1 Introducció

Aquest capítol està estructurat de manera que primer s'explica la definició de cada tecnologia que s'ha utilitzat. Després en posteriors apartats s'explica com participa cada tecnologia a la implementació de la solució.

8.2 Tecnologies i eines utilitzades

8.2.1 Programació Orientada a Aspectes

La Programació Orientada a Aspectes, en anglès Aspect Oriented Programming o bé AOP[1], és una metodologia de programació força recent.

L'AOP és un paradigma de programació orientat a aïllar en entitats ben definides funcionalitats secundàries, extra-funcionals o supletories d'una lògica de negoci implementada en un llenguatge d'alt nivell[FIGURA crosscut.jpg]. Ajuda, mitjançant la separació de conceptes transversals, a augmentar la modularitat del programa natiu.

En un programa s'enten com a concepte com un conjunt de comportaments o accions que aquest utilitza per a dur a terme una finalitat o requeriment. De vegades part d'aquest conjunt d'accions es repeteixen en el codi per solucionar diferents requeriments. La repetició d'aquest bloc de codi arreu del programa natiu és el que s'anomena concepte transversal. Sense tenir

[1]AOP - http://es.wikipedia.org/wiki/Programación_Orientada_a_Aspectos

en compte tecnicismes d'informàtica es podria anomenar aquests conceptes transversals com certs aspectes de l'aplicació que es repeteixen al llarg del codi i d'aqui prové el nom d'*Aspect-Oriented Programming*.

La materialització de l'anteriorment mencionada entitat s'anomena aspecte, en anglés (*aspect*).

Tot i que mitjançant l'AOP i a partir d'un codi d'alt nivell mínim es poden resoldre aspectes funcionals de problemes complexos, o fins i tot la totalitat del problema, no està acceptat utilitzar-la per aquesta finalitat perquè no estaria justificat l'increment de complexitat que l'AOP introduïria.

Un desavantatge important que comporta l'ús de la metodologia AOP en una aplicació és que es fa més complex el seu desenvolupament, ja sigui en termes de configuració o per falta de coneixemnts del programador, però el ventall de beneficis que comporta és força gran:

- Permet resoldre de forma clara, senzilla i eficaç problemes que en la capa de llenguatge d'alt nivell es ressoldrien mitjançant blocs de codis duplicats o repartits entre una arquitectura de dades complexa.
- En conseqüència del punt anterior el codi és més fàcil de mantenir.
- En part, gràcies també al primer punt, en el cas de conseguir reduïr una arquitectura de dades complexa en un sol aspecte, s'aconsegueix desacoplar de manera significativa el codi d'alt nivell. Un cop més, fent-lo més fàcil de mantenir o extendre'n les funcionalitats.

A continuació es defineixen els termes més importants de l'AOP, en anglès per la seva claretat. Entre ells els mencionats en el paràgraf anterior:

- **Aspect**
  Representen la solució als conceptes transversals que s'han mencionat prèviament. Abans de programar en AOP s'ha d'identificar quins 'aspectes' (en el sentit literal de la paraula) de l'aplicació es volen tractar. Un aspecte pot tenir un o més *advices* i un o més *pointcuts*, ambdós termes explicats en breu.
- **Joinpoint**
  Representa el punt d'execució dins d'un programa el qual un aspecte

interceptarà. Per exemple, la crida a un mètode o l'assignació d'una variable.

- **Pointcut**
  Defineix el *joinpoint* mitjançant una condició a complir. Aquesta definició es fa mitjançant expressions regulars, i gràcies a aquestes, és possible definir en un sol pointcut més d'un joinpoint. Mitjançant el pointcut es produïran les intercepcions als *joinpoint* i, un cop interceptats, s'executaran els consells o *advice*. Aquestes regles definides doncs tenen com a finalitat identificar crides de funcions, noms de variables, tipus de variables, quantitat de paràmetres, assignacions de valors, en definitiva, la majoria de les funcionalitats que permeti la definició del llenguatge d'alt nivell sobre la que treballa l'AOP, sempre i quan, la implementació concreta de l'AOP també ho permeti. ...

- **Advice**
  S'anomena consell o *advice* al codi que es vol aplicar sobre el codi existent quan es compleixi un *pointcut*, és a dir, quan hi hagi una ocurrència o 'match' en les definicions establertes en el *pointcut*. Existeixen implementacions d'AOP, com AspectJ que s'explica en el següent subapartat, que permeten concretar d'una manera encara més profunda quan aplicarà el point cut que el connecta: abans de la crida al mètode, durant la seva execució o a la finalització del mètode.

- **Weaving**
  Com s'ha mencionat anteriorment, l'AOP està per sobre del llenguatge d'alt nivell. Degut a que el codi que te per sota és indispensable pel seu funcionament, el codi generat en AOP és compilat i injectat en el codi de l'aplicació la qual es desitja aspectuar. Aquest codi generat és dependent del llenguatge que hi ha per sota i això implica la necessitat d'un compilador d'AOP per a cada llenguatge d'alt nivell existent.

- A aquest procés d'injecció se l'anomena *Weaving* i n'existeixen de dos tipus:

  - *Load Time Weaving*
    Es quan el procés de *weaving* es realitza en temps d'execució mitjançant el *ClassLoader* utilitzat per a carregar les classes. Aquest procés no es pot utilitzar en dispositius mòbils per culpa de la divisió del procés de verificació.

– *Compile Time Weaving*
  Es quan el procés de *weaving* es realitza en temps de compilació. Aquest procés es mostra gràficament en la figura 8.1.

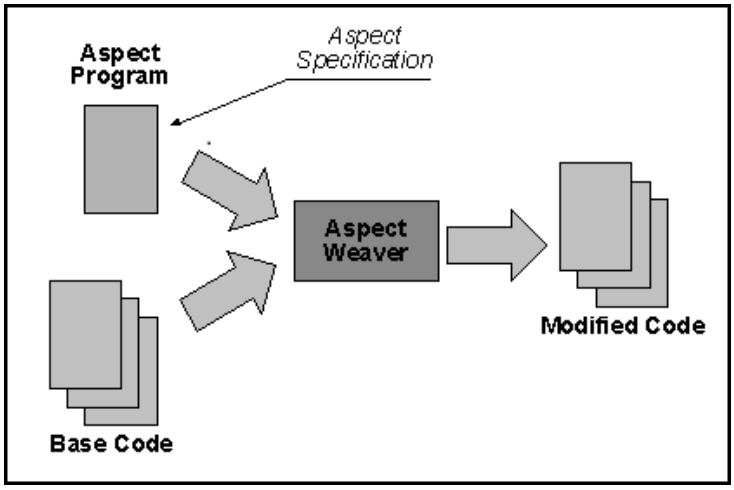

Figura 8.1: Funcionament del procés de weaving

Es pot veure com interactuen tots aquests termes en la figura 8.2.

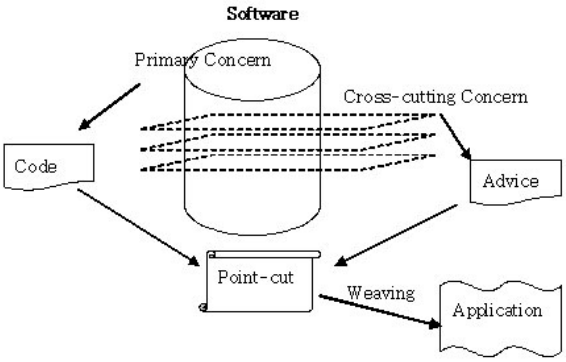

Figura 8.2: Funcionament d'AspectJ

Per clarificar tots els punts anteriors es proposa un exemple molt senzill: donada una aplicació implementada en Java, es vol interceptar i llençar un missatge per consola cada cop que l'aplicació llença un missatge per consola mitjançant el mètode: `{System.out.println(String)}`.
Seria el cas del concepte a tractar.

Un cop identificat el concepte, es procedeix a implementar un aspecte que, com s'ha mencionat anteriorment, contindrà en aquest cas un *pointcut* i un consell:

- **Pointcut**
  Mitjaçant les expressions regulars que permeti la sintaxi de la implementació d'AOP, en aquest exemple AspectJ, es defineix el *pointcut* següent: tota crida durant l'execució de l'aplicació a un mètode anomenat `{println (..)}` s'anomena captura.

- **Advice**
  Per cada cop que es compleixi la regla anterior s'executarà el codi contingut dins del consell. En aquest exemple es vol que s'executi en el moment abans de la crida.

La part en la que s'ha de tenir més cura és en la definició del *point cut*, ja que un error aquí pot dur a terme en un comportament inesperat del concepte transversal tractat o, fins i tot, de l'aplicació aspectuada en el cas que des de l'aspecte se'n modifiquès el funcionament.

En l'anterior exemple, s'ha interceptat qualsevol crida a un mètode println(String) i això inclou tant les crides a la funció del sistema sistema out.println(String), com possibles mètodes definits amb el mateix nom. Així doncs aquesta també és la part més característica de l'AOP ja que demostra que es pot interceptar qualsevol mètode de l'aplicació que es té per sota.

Un cop implementada la solució AOP es compilaria i s'injectaria al codi Java prèviament compilat via weaving. En la següent imatge es mostra el codi utilitzat per aquest exemple.

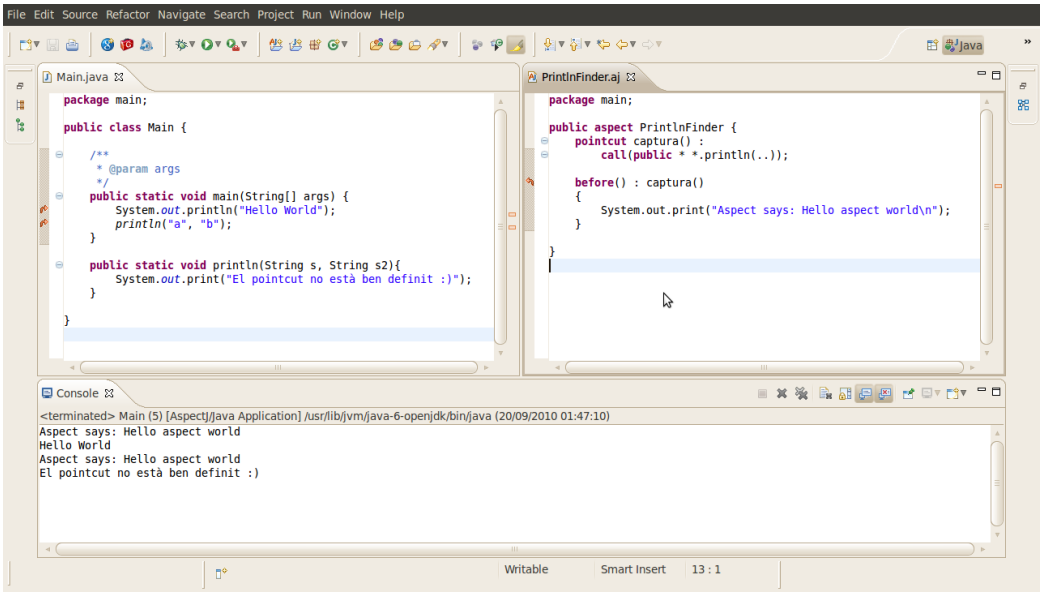

Figura 8.3: Procés de weaving

8.2.2 AspectJ

[10][9] *AspectJ*[2] és una implementació per a Java d'AOP. És un projecte existent al públic des de l'any 2001 creat inicialment per PARC i actualment desenvolupat per la fundació Eclipse. El fet de que la seva sintaxis estigui basada en Java el fa un llenguatge simple d'entendre almenys a nivell tècnic tenint en compte que l'utilitzen programadors Java. És lògic, però existeixen altres implementacions que tot i que la funcionalitat del llenguatge és la mateixa, la seva escriptura dista molt del la del codi d'alt nivell.

Al ser un projecte desenvolupat per la fundació Eclipse, aquesta ha integrat l'AspectJ en algunes de les versions del seu IDE, l'Eclipse, brindant així al programador interessants utilitats com vistes de configuració, editors avançats que permeten a partir d'un *joinpoint*, accedir al point cut que el defineix, un compilador integrat, i moltes més opcions. També és possible instal·lar-lo a l'IDE en forma de plugin.

[2]AspectJ - http://www.eclipse.org/aspectj/

8.2.3 Google Web Toolkit

[11] *Google Web Toolkit*[3], o GWT, és un framework creat per google per a facilitar la creació de llocs web dinàmics encapsulant la tecnologia AJAX[4], Asyncronus JavaScript And XML. La web es programa íntegrament en Java, estructurant el codi d'una manera definida pel framework (seguint el principi COC, *Convention Over Configuration*[5]). El compilador que inclou s'encarrega de generar codi HTML + JavaScript que serà el resultat final pel navegador. Un projecte GWT es divideix en mòduls. Cada mòdul inclou codi, recursos web i arxius de configuració.

El codi s'estructura en dos blocs:

- **El bloc servidor**
  Constarà de Servlets i demés classes Java, molt similar a qualsevol aplicació Web basada en Java

- **El bloc client**
  També estarà implementat íntegrament en Java. Aquest bloc defineix la interfície d'usuari mostrada al navegador. Conceptualment, s'allunya molt d'una aplicació web ordinària ja que gaudeix de moltes més similituds amb el desenvolupament d'interfícies gràfiques per a aplicacions d'escriptori. Per tal de generar la interfície gràfica es fan anar els elements gràfics proporcionats pel framework GWT. Un cop compilat, tot aquest bloc esdevindrà Javascript, gràcies tant a les eines de GWT com a la implementació pròpia de les llibreries de Java que incorpora. Degut a a l'ús d'aquestes llibreries el conjunt de classes que es poden utilitzar és limitat respecte a J2SE (només es poden utlitzar les classes per a les que GWT proporciona implementació). Així aquest bloc serà executat al navegador.

Quant a arxius de configuració, els més importants són el web.xml, que funciona exactament com qualsevol altra aplicació web desenvolupada en Java, i el descriptor del mòdul. Aquest fitxer és del tipus XML, i determina les propietats del mòdul, tals com la dependència d'altres mòduls o el punt d'entrada.

[3]GWT - http://code.google.com/intl/es-ES/webtoolkit/
[4]AJAX - http://es.wikipedia.org/wiki/AJAX
[5]COC - http://en.wikipedia.org/wiki/Convention_over_configuration

Cada mòdul ha de tenir com a mínim una classe que extengui la classe *EntryPoint*. Aquesta classe pot ser especificada llavors al descriptor del mòdul com a punt d'entrada i serà la classe que s'executarà inicialment al navegador, ja que, a l'estar definida al descriptor com a punt d'entrada, serà automàticament cridada pel Javascript que apareix per defecte a la plana html inicial.

L'últim concepte, però no el menys important, és la comunicació entre classes del bloc client i el bloc servidor. En el cas que no fos necessària aquesta comunicació es parlaria de webs estàtiques, i el bloc de codi del servidor estaria buit. Tot i que possible, no tindria massa sentit llavors l'elecció de GWT per al desenvolupament de l'aplicació, ja que resultaria més senzill un disseny basat exclusivament en html i css.

La comunicació en GWT és resolta mitjançant peticions (Remote Procedure Call's) asíncrones des del codi del bloc client cap a un servei. En realitat un servei consta de 2 interfícies i una classe. Suposem un servei anomenat GestioUsuaris:

- **GestioUsuarisService**
  Seria la interfície que només indicaria quins mètodes estàn disponibles per les crides RPC i quins objectes retornaran.
- **GestioUsuarisServiceAsync**
  Ha d'estar definida al bloc client, seria la interfície que utilitzarà el codi del bloc client per a fer peticions. Els seus mètodes correspondran exactament als de la interfície *GestioUsuarisService*, amb la única diferència que cada mètode retornarà null i tindrà un paràmetre extra que serà un callback, el codi del qual serà cridat quan la petició s'hagi resolt satisfactòriament.
- **GestioUsuarisServiceImpl**
  Serà la classe, definida al bloc servidor. Extendrà *RemoteServiceServlet* (classe proporcionada pel propi GWT) i implementarà la interfície *GestioUsuarisService*.

Els paràmetres emprats als mètodes del servei, així com els objectes que poden retornar els mètodes, poden ser qualsevol tipus primitiu de Java o bé qualsevol objecte Java que sigui suportat per la implementació de la llibreria Java de GWT, així com qualsevol objecte Java definit al bloc client que pugui ser serialitzat.

8.2.4 Altres tecnologies i eines

- **Ant**
  Apache Ant[6] és una eina de software utilitzada per automatitzar processos de compilació i construcció. Funciona definint les tasques a realitzar en un fitxer XML i, ja sigui des de consola o bé programàticament, se li va especificant quines tasques es desitja que realitzi. Per tal d'utilitzar-lo programàticament resulta fàcil d'integrar-lo en projectes Java ja que ha estat escrit amb aquest mateix llenguatge. La seva llicència és Apache License 2.0.

- **Drools**
  Drools[7] és un conjunt d'eines desenvolupades per la JBoss Community. Es tracta d'una plataforma de Business Logic Integration, de la qual només s'ha utilitzat Drools Expert en aquest projecte. Drools Expert és un motor d'inferència basat en regles desenvolupat íntegrament en Java. El fet que sigui fàcilment integrable en projectes Java i, sobretot, que permeti l'ús de Java Beans com a base de coneixement han fet que es descartés Jess, el motor d'inferència basat en regles que es pensava utilitzar inicialment per al component servidor.

- **Tomcat Apache**
  Tomcat[8] és un famós servidor d'aplicacions open source creat i mantingut per la fundació Apache. La seva llicència és Apache License version 2.

- **Wireless Universal Resource FiLe**
  Durant el transcurs d'aquest document cada cop que s'ha fet referencia a WURFL[9], *Wireless Universal Resource FiLe*, s'ha anomenat base de dades de perfils de dispositiu. WURFL no és res més que un fitxer XML on hi són perfectament definits tots aquests perfils. Es tracta d'un projecte open source i això implica que aquest fitxer sigui actualitzat cada cert temps gràcies al fort recolzament de la comunitat vers aquesta eina. Des del propi portal web, es poden descarregar llibreries per diversos llenguatges de programació com poden ser Java, .NET o PHP.

- **JDBC**
  Java Data Base Connectivity[10] és una API que ofereix Java per a la

[6] Ant - http://es.wikipedia.org/wiki/Ant
[7] Drools - http://jboss.org/drools/
[8] Tomcat - http://tomcat.apache.org/
[9] WURFL - http://wurfl.sourceforge.net/
[10] JDBC - http://es.wikipedia.org/wiki/Java_Database_Connectivity

conexió amb bases de dades que cumpleixin amb els estandards ODBC. En aquest projecte s'ha fet anar el MySQL JDBC Connector.

- **JDOM**
  Java Document Object Model[11] és una llibreria de Java per a gestionar contingut XML programàticament.

- **Llibreries d'Apache Commons**
  Durant el desenvolupament del sistema, per a dur a terme certs requisits funcionals ha sorgit la necessitat d'incloure com a dependències, sobretot al servidor, diverses llibreries d'*Apache commons*[12]. Entre elles estan, per exemple, FileUpload, BeanUtils, IO o Collections.

- **XML**
  eXtensive Markup Language és un meta-lleguatge especificat pel W3C. El seu objectiu és codificar informació de forma que aquesta pugui ser entesa fàcilment per un sistema informàtic. Actualment és una tecnologia molt utilitzada.

- **DTD**
  S'ha utilitzat aquesta tecnologia, *Document Type Definition*[13], per al correcte funcionament del pas de missatges entre servidor i client. Serveix per a definir la estructura de contingut xml.

- **MySQL**
  MySQL[14] és un gestor de base de dades relacional amb llicència GNU GPL. Segons wikipedia n'existeixen almenys sis milions d'instal·lacions.

- **CVS**
  Concurrent Versions System[15], és una eina de lliure distribució orientada a portar un control de versions de codi font. S'ha emprat durant tot el desenvolupament del projecte per tal de desenvolupar-lo col·laborativament i de forma segura i ordenada.

- **Google Docs**
  Famosa eina[16] de Google que s'ha emprat tant per a dur a terme la documentació del projecte com per a portar una mena de seguiment

[11] JDOM - http://www.jdom.org/
[12] Apache commons - http://commons.apache.org/
[13] DTD - http://es.wikipedia.org/wiki/Definición_de_tipo_de_documento
[14] MySQL - http://www.mysql.com/
[15] CVS - http://es.wikipedia.org/wiki/CVS
[16] Google Docs - http://docs.google.com

d'errors durant el procès d'implementació. Es tracta d'una suite d'ofimàtica amb propietats col·laboratives molt interessants.

- **Llistes de distribució**
  S'ha fet ús de llistes de distribució de les comunitats de Google Web Toolkit, J2ME i AspectJ.

- **Eclipse**
  És un entorn de desenvolupament integrat (IDE) que s'ha utilitzat per desenvolupar tots els mòduls que composen el component client. Està desenvolupat íntegrament en Java tot i que, al utilitzar SWT com a llibreria gràfica, és depenent de plataforma. Està disponible mitjançant la llicència Eclipse Public License[17].

- **Netbeans**
  És un entorn de desenvolupament integrat (IDE) que s'ha utilitzat per desenvolupar tant el component servidor com la web. Està desenvolupat per Sun Microsistems (ara Oracle) íntegrament en Java.

- **GWT4NB**
  És un plugin per a Netbans que ofereix suport per al desenvolupament d'aplicacions web mitjançant el framework Google Web Toolkit.

- **Latex**
  S'ha utilitzat aquest sistema de composició de documents basats en TeX. És open source i la seva llicència és Licencia Pública del Proyecto LaTeX LPPL.

- **Dia**
  Dia és un editor de diagrames open source i s'ha utilitzat per a la creació dels diferents gràfics d'aquest document. La seva llicència és GNU GPL.

- **MTJ**
  MTJ o Mobile Toolkit for Java és un plugin per a Eclipse que facilita el desenvolupament per a dispositius mòbils.

- **AJDT**
  AJDT o AspectJ Developer Tools és un plugin per a Eclipse que proporciona al desenvolupador diverses eines, un compilador i la pròpia llibreria AspectJ, per a facilitar el desenvolupament amb aspectes en la plataforma Java.

[17]Eclipse Public License - http://www.eclipse.org/legal/epl-v10.html

- **WTK**
  El Wireless ToolKit seria pels dispositius mòbils el que Java SDK és per als sobretaula. És un conjunt de llibreries, eines i emulador distribuits per Sun que permeten utilitzar el llenguatge Java (J2ME) per al desenvolupament d'aplicacions en dispositius mòbils. Aquest es distribueix tant per al desenvolupament en plataformes Linux com Windows.

- **Google Project Hosting**
  S'ha utilitzat aquesta eina[18] de Google per a la documentació millorant les capacitats de google docs.

- **SVN**
  Es un dels sistemes de control de versions que ofereix google per a gestionar el codi font dels seus projectes. S'ha instal·lat el seu client al Netbeans.

8.3 Detalls de la implementació

A continuació s'explica finalment com s'ha dut a terme la solució en termes d'implementació. Primer s'explica el client i després el servidor.
Durant tot el procés de desenvolupament s'ha intentat seguir l'estandard de les Java Naming Conventions, és a dir, escrivint segons aquestes tots els noms dels beans, atributs, mètodes, variables i clases.

8.3.1 Client

El client, anomenat fins ara component client, és la part de software que s'injectarà a l'aplicació a adaptar. S'ha optat per una implementació íntegra en Java, però degut a que es va pensar en donar suport tant a aplicacions J2SE i J2ME en una única solució, s'ha hagut de fer un disseny modular per tal de poder reciclar el màxim de codi possible.

Aquest disseny modular consta de 3 capes o tipus de mòdul:

- ***.common**
  Inclou la implementació de la lògica principal de la solució, així com les interfícies que hauran d'implementar les altres capes.

[18] Google Project Hosting - http://code.google.com/hosting/

- ***.platform**
  A aquesta capa s'implementen les interfícies proporcionades per la capa comuna que són depenents de plataforma. Hi haurà doncs un mòdul d'aquest tipus per cada plataforma suportada al sistema (actualment hi han dos implementats: un per a J2ME i un altre per a J2SE).

- ***.ui**
  Aquesta és la capa que es vincula amb les llibreries gràfiques de l'aplicació mitjançant AOP. Un mòdul d'aquesta capa ha d'incloure la capa aspectual necessària, els mecanismes de selecció de ProposalSet i obtenció de feedback així com especificar quines implementacions de les disponibles han de ser utilitzades. Hi haurà doncs una implementació d'aquesta capa per a cada llibreria gràfica a la que es vulgui donar suport. S'han implementat els mòduls per a LWUIT i LCDUI (J2ME) i AWT, SWT i Swing(J2SE).

Capa *.common

Compatibilitat de compilació amb l'entorn d'execució de CDC-1.1/Foundation-1.1 Com s'ha mencionat en el capítol de Java la solució existeix tant per entorn d'escriptori com per a dispositius mòbils. Aquests últims però tenen limitacions de processament i per tant tenen un joc d'instruccions molt més reduït. Llavors la solució passa per obligar al desenvolupador a implementar un codi que compili amb l'entorn d'execució CDC 1.1.

N'hi ha hagut prou doncs, en crear un projecte Java, i marcar l'opció a les propietats de projecte. Malauradament pel fet de fer el projecte compatible amb CDC i no haver-hi l'opció que també ho sigui per CLDC, no hi ha restricció a nivell de compilador a l'hora de crear codi per CLDC i el programador ha de ser concient del codi que hi posa ja que pot no funcionar en dispositius que utilitzin CLDC.

Les restriccions CDC impedeixen, per exemple, utilitizar qualsevol de les funcionalitats del java 5 com ara els Generics, o també qualsevol implementació de List. Totes aquestes limitacions han fet d'aquest projecte una mica més complex d'implementar.

Fitxer de propietats Aquest fitxer és generat pel component Web després de crear una adaptació i consta d'una serie de metadades, tuples clau=valor, essencials per al correcte funcionament del component client i del sistema en general.

Els més importants són:

- **userId**
  És l'identificador única d'un component client concret.
- **serverURL**
  És la cadena de conexió amb el servidor.
- **connectionType**
  Indica quin protocol de conexió utilitzarà el dispositiu.
- **persistenceType**
  Indica quin protocol de persistència utilitzarà el dispositiu.
- **uiType**
  Indica quina llibreria gràfica utilitzarà el dispositiu.
- **feedback**
  Indicador booleà sobre la presència de la pantalla de satisfacció o no.
- **feedbackRank**
  Rang màxim en l'escala de la pantalla de satisfacció.

Interfícies que han de ser proporcionades per les altres capes Aquest mòdul necessita de certes classes que inexorablement dependran de plataforma. Aquestes classes seran proporcionades per les altres capes del component client, però la seva definició (la interfície que implementen, parlant de Java) ha de ser definida aquí. Les interfícies que es presenten a la resta de capes són les següents:

- **Server**
  Aquesta intefície defineix com el component client interactuarà amb el component servidor.
- **Persistence**
  Aquesta interfície defineix una cache local, que utilitzarà el component client per emmagatzemar logs fins que no s'enviïn. Aquest pas és necessari ja que sempre pot haver-hi problemes de connectivitat.

- **Transformer**
  Aquesta interfície s'encarrega de transformar les dades obtingudes pel servidor a Java Beans. Actualment només hi ha una implementació d'aquesta interfície (XmlTransformer), que és proporcionada per aquest mateix mòdul. Per a més informació consultar Desacoblar l'XML, abstraient el protocol de comunicació, al capítol 11: Conclusions i Treball Futur.

- **UIAdapter**
  Aquesta interfície treballa directament amb objectes de la interfície d'usuari. A partir d'un ProposalSet determina si els objectes de la interfície que li envia la capa aspectual han de ser adaptats o no i, en cas afirmatiu, aplica els canvis pertinents.

- **ConcreteFactory**
  Aquesta interfície proporcionarà les classes concretes que implementen les interfícies anteriorment esmentades. Quan el component s'inicia, el primer pas a fer és inicialitzar els controladors del mòdul comú amb la ConcreteFactory proporcionada al mòdul de les llibreries gràfiques, així el mòdul comú ja pot obtindre les implementacions que necesita pel seu funcionament.

Controladors Aquest mòdul té dos controladors, que són els que implementen tota la lògica del component client comuna per a totes les plataformes. Els controladors són inicialitzats i utilitzats directament per la capa aspectual.

Els controladors són:

- **LogController**
  Gestiona tota la part d'emmagatzematge i tractament la informació capturada pels aspectes, tant sobre la captura de la interfície d'usuari com sobre les accions de l'usuari sobre la mateixa.

- **ProposalController**
  Gestiona el tractament de propostes i proporciona la implementació de UIAdapter a qui la necesiti.

Capa *.platform

A diferència de l'anterior aquest projecte pot utilitzar qualsevol versió/compilador de Java SDK sempre i quan en el dispositiu existeixi un entorn d'execució

compatible.
Ha d'implementar únicament les interfícies proporcionades per la capa comuna que són depenents de plataforma. Aquestes interfícies són Server i Persistence. Cal tindre en compte, però, que un mòdul d'aquesta capa pot proporcionar diverses implementacions per a la mateixa interfície, deixant a l'implementador de la ConcreteFactory l'elecció de quina de les implementacions es vol utilitzar en un cas concret.
Actualment s'han proporcionat dos projectes diferents, és a dir, s'ha implementat suport per a dues plataformes:

- **J2ME**
  Proporciona una implementació de Persistence: RMSManager. Aquesta classe es basa en la API RMS javax.microedition.rms de J2ME per a l'emmagatzemament de dades en persistència local del dispositiu.
  Proporciona una implementació de Server: HttpServer. Aquesta classe es basa principalment en la classe javax.microedition.io.HttpConnection (proporcionada per CDC) per a oferir un canal de comunicació HTTP amb el component servidor.
- **J2SE**
  Proporciona una implementació de Persistence: FastFSManager. Aquesta classe empra el sistema de fitxers del sistema per tal d'emmagatzemar dades.
  Proporciona dues implementacions per a Server:
  - *DesktopHttpServer*
    Basada en HttpClient (llibreria proporcionada per Apache Commons)
  - *FakeDemoServer*
    Implementació creada amb motius de depuració i desenvolupament. Aquesta implementació de Server no es comunica amb un component servidor, sinó que l'emula. Per tal d'emular-lo fa veure que envia dades (que no són tractades ni emmagatzemades enlloc) i envia una resposta que obté directament del sistema de fitxers. Així el desenvolupador pot controlar la resposta que s'envia.

Capa *.ui

L'última capa del component client és la única que conté tecnologia AOP. Aquesta capa proporciona doncs l'enllaç entre l'aplicació que es vol adaptar

i els controladors implementats a la capa comuna, a més de tot el que depengui explícitament de la llibreria gràfica que utilitza l'aplicació, com ara la pantalla de selecció de proposal set o bé la de feedback. Per tal d'ajudar al desenvolupament es va decidir separar els aspectes en tres tipus ja que conceptualment s'utilitzen per a tres funcions ben diferenciades:

- **Capturar el cicle d'aplicació**
  Aquest aspecte té la missió de detectar quan s'ha iniciat l'aplicació per a inicialitzar el component client. A més ha de capturar també quan es vol tancar l'aplicació i comunicar-ho als controladors, per si tenen que fer alguna tasca abans de deixar d'existir (com tancar canals de comunicació, finalitzar correctament el procés persistència, etc).
  Finalment també poden tindre en compte situacions de canvi, com que al mòbil rebem una trucada o que a l'escriptori es minimitzi l'aplicació que s'està monitoritzant. Aquests events afecten al cicle de l'aplicació, per exemple, a l'hora de determinar el temps d'ús d'una finestra o el temps de resposta d'un usuari.

- **Capturar la interfície gràfica**
  Aquest aspecte s'ha d'encarregar de capturar la creació i destrucció de finestres a l'aplicació, i emmagatzemar els elements que les formen amb els Beans que ofereix la capa comuna. Un cop les finestres es volen destruir ha de comunicar-ho al LogController, que s'encarrega de monitoritzar el temps que han estat obertes i desar-les apropiadament. A més a més quan es crea una nova finestra o un nou element a la interfície aquest aspecte és l'encarregat d'aplicar-li els canvis descrits al proposal set.

- **Capturar les accions d'usuari**
  Aquest aspecte s'ha d'encarregar de capturar tota interacció de l'usuari amb la interfície gràfica.

A més de proporcionar la capa aspectual i les pantalles de selecció de ProposalSet i de feedback, aquesta capa també té la responsabilitat de determinar quines classes s'han d'utilitzar per a satisfer les interfícies de la capa comuna. Així doncs aquesta capa ha de proporcionar una implementació de la interfície ConcreteFactory, tot retornant una instància de la classe que es vulgui utilitzar, mitjançant el fitxer de propietats, per a la interfície que representa cada mètode.

S'han implementat els següents projectes que proporcionen aquesta capa:

- ***.j2me.lcdui**
  Suporta la llibreria gràfica LCDUI, que utilitzen algunes aplicacions J2ME. Cal destacar que aquesta llibreria és molt bàsica i per tant la monitorització és molt limitada.

- ***.j2me.lwuit**
  Suporta la llibreria gràfica LWUIT, per a J2ME.

- ***.j2se.awtSwing**
  Suporta tant AWT com Swing en entorns J2SE. Ha estat programada amb Java 6 així que no funcionarà a entorns amb un JRE de versió inferior encara que l'aplicació que es vol adaptar estigui desenvolupada amb una versió anterior de Java.

- ***j2se.swt**
  Suporta SWT per a entorns J2SE que disposin d'implementació d'aquesta llibreria. Com que SWT no és multiplataforma aquest mòdul utilitza Swing per a les pantalles de selecció de propostes i la de feedback.

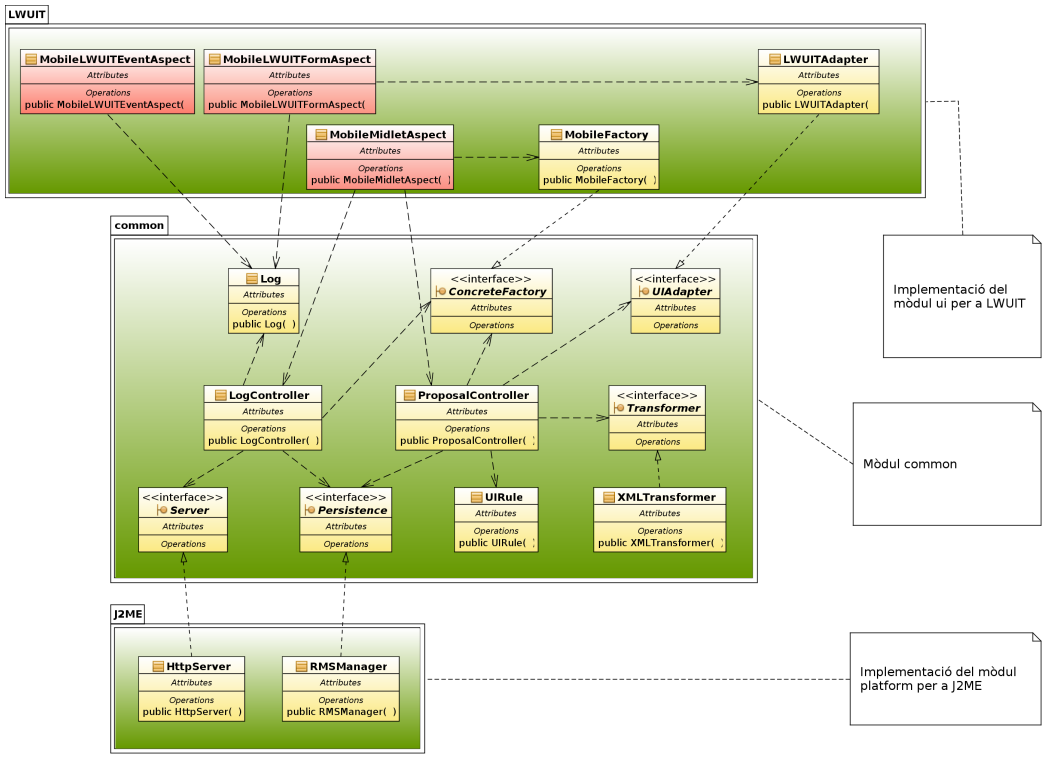

Figura 8.4: Diagrama de dependència del component client implementat per a LWUIT

8.3.2 Servidor

Al capítol anterior es parlava del component servidor i de l'aplicació web com a dos components completament separats. Des del punt de vista de la funcionalitat això és cert. A l'hora de proporcionar una implementació, però, té molt més sentit la seva fusió, ja que s'ha optat per implementar-los amb les mateixes tecnologies. La web ha sigut implementada mitjançant GWT, que utilitza Java Servlets mentre que al component servidor s'ha optat per emprar un canal HTTP per a la comunicació amb el client. Per implementar aquest canal, doncs, s'ha decidit utilitzar també Java Servlets i reduir així l'impacte que suposaria utilitzar altres tecnologies.

Web

Degut a que s'ha decidit no optar per cap capa de més alt nivell a l'hora de treballar amb GWT s'han hagut d'utilitzar alguns patrons i convencions per tal de facilitar la feina el màxim possible. Aquestes facilitats són les següents:

- **Patró Model-Vista-Controlador**
  S'ha optat per utilitzar el patró MVC per a la implementació de la part dinàmica de la interfície d'usuari. Així cada pantalla de la web es divideix en tres components:

  - **Vista (arxius *view.java)**
    Defineix, mitjançant els components que ofereix el framework GWT, la interfície d'usuari. Totes les vistes extenen la classe FlowPanel, proporcionada pel GWT, que esdevé un element DIV un cop compilada.
  - **Model (arxius *model.java)**
    Emmagatzema l'estat de la vista, així com les dades que aquesta necesiti mostrar.
  - **Controlador (arxius *controller.java)**
    Proporciona tota la lógica de la vista. Això pot incloure demanar dades al servidor per actualitzar el model, enviar dades del model al servidor pel seu emmagatzemament i validació de les dades que l'usuari insereix a la vista.

- **Patró Observer**
  Per tal que la vista s'assabenti quan hi ha hagut un canvi al model es fa ús del patró Observer. S'ha definit doncs una interfície anomenada Observer, que és implementada per les vistes. Aquesta interfície consta

d'un sól mètode, anomenat update, que és cridat pel model quan aquest rep algun canvi. Per tal que el model pugui tractar Observers, s'ha implementat la classe abstracta Observable, que defineix mètodes per registrar Observers i avisar-los quan sigui necesari. El llenguatge Java ofereix un patró similar inclós a les seves llibreries. S'ha optat per una implementació pròpia, però, degut a que aquest codi esdevindrà Javascript un cop compilat, i la implementació que ofereix Java no és compatible actualment amb el compilador Javascript de GWT.

- **SoftCallback**
  Com s'ha mencionat anteriorment, per tal de poder fer una crida RPC (necesària per enviar o demanar dades al servidor), es precisa implementar un callback, que s'executarà quan la crida s'hagi resolt, ja sigui satisfactòriament o no. GWT proporciona una interfície, anomenada AsyncCallback, que consta de dos mètodes: onFailure i onSuccess. Per tal de no repetir codi i oferir un comportament comú davant un error en la crida RPC (com pot ser la pèrdua de connectivitat, per exemple) s'ha implementat la classe abstracta SoftCallback, que implementa el mètode onFailure d'AsyncCallback.

- **AbstractService**
  Es tracta d'un Service abstracte, que implementa mètodes bàsics que requereixen la gran majoria de Services, com ara l'obtenció de l'usuari autenticat al sistema actualment, etcètera.

- **ProjectBuilder**
  El ProjectBuilder és el mòdul de software capaç de compilar aplicacions Java. S'ha implementat de forma genèrica, i fa ús del projecte Apache Ant. Aquest mòdul s'utilitza per a injectar de forma automàtica el component client a les aplicacions que es volen adaptar. Requereix de certes dades, però, per a oferir l'aplicació adaptada. Aquetes dades són: el codi font de l'aplicació a adaptar, el codi font dels mòduls del component client que es vol injectar, diverses dades necesàries per a la compilació, que depenen de la plataforma (J2ME ó J2SE, actualment), el build.xml per a la plataforma seleccionada i la ID de l'adaptació. Totes aquestes dades existeixen a la base de dades.

- **JDBCUtil**
  Aquesta classe proporciona mètodes per a simplificar l'accés a la base de dades. Encapsula totes les crides JDBC i permet als Service treballar directament amb objectes del tipus CachedRowSet.

Component servidor

El component servidor disposa d'un Servlet, anomenat LogReceiver, que s'encarrega d'esperar logs. Per cada log que rep, retorna un proposal set al client que li ha enviat. S'ha decidit però deixar que aquesta classe només s'encarregui de la lògica del procés, i actuï a nivell de controlador. Per tal d'emmagatzemar el log rebut i obtindre un proposal set que retornar LogReceiver fa ús de la classe LogUtils. Tanmateix LogUtils es comunica amb la base de dades (tant per emmagatzemar el log com per a obtindres les dades a passar al motor d'inferència) a través de la classe JDBCUtil, anomenada a l'apartat anterior. A més genera respostes a través d'una interfície anomenada InferenceEngine, que representa una capa d'abstracció respecte al motor d'inferència concret que es vol utilitzar.
Es proporciona una implementació de InferenceEngine que utilitza Drools com a motor d'inferència, a la classe DroolsInferenceEngine.

8.3.3 Comunicació bidireccional client-servidor

Per tal de que la comunicació a dos bandes entre el client i servidor sigui efectiva i a prova d'errors s'ha decidit estabir un protocol de comunicació basat en XML. Per tal de validar aquest XML tant en el client com en el servidor es dissenya un Document Type Definition, DTD.

A l'existir dos tipus de comunicacions, d'una banda client-servidor i per l'altra servidor-client, amb funcionalitats clarament diferents s'ha generat un DTD per a cada cas.

- **Client-servidor**
  El component client és l'encarregat de recollir en forma de logs el contexte d'ús entre l'usuari i l'aplicació. Aquests logs en XML són creats en el component client complint en el següent DTD:

```
 <!DOCTYPE MESSAGE [
<!ELEMENT MESSAGE (LOG+)>
<!ELEMENT LOG (HEAD,BODY)>
        <!ATTLIST LOG    ID        ID        #REQUIRED>
<!ELEMENT HEAD (USER,DEVICE,DATE,APPLICATION)>
<!ELEMENT USER (#PCDATA)>
<!ELEMENT DEVICE (#PCDATA | (PROPERTY+))>
<!ELEMENT PROPERTY ID (#PCDATA)>
        <!ATTLIST PROPERTY ID    ID        #REQUIRED>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT APPLICATION(NAME,VERSION,TYPE,JAVA)>
<!ELEMENT NAME (#PCDATA)>
```

```
<!ELEMENT VERSION (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT JAVA (#PCDATA)>
<!ELEMENT BODY (FRAME+)>
<!ELEMENT FRAME (TIME,CONTAINER,EVENTS+)>
          <!ATTLIST FRAME ID ID #REQUIRED>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT CONTAINER (LAYOUT,TYPE,WIDGET+,CONTAINER+)>
<!ELEMENT LAYOUT (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT WIDGET (HSIZE, VSIZE ,FOREGROUND_COLOR,
        BACKGROUND_COLOR,TYPE,FONT)>
                  <!ATTLIST WIDGET ID ID #REQUIRED>
<!ELEMENT HSIZE (#PCDATA)>
<!ELEMENT VSIZE (#PCDATA)>
<!ELEMENT FOREGROUND_COLOR (#PCDATA)>
<!ELEMENT BACKGROUND_COLOR (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT FONT (TYPE, SIZE ,STYLE)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT SIZE (#PCDATA)>
<!ELEMENT STYLE (#PCDATA)>
<!ELEMENT EVENTS (GAIN_FOCUS, LOST_FOCUS,
          ACTION_PERFORMED)*>
<!ELEMENT GAIN_FOCUS (TIME)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT LOST_FOCUS (TIME)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT ACTION_PERFORMED (TIME)>
<!ELEMENT TIME (#PCDATA)>
]>
```

No s'explica tot el DTD sencer, però s'explicarà els trets més importants que tenen a veure amb tot el que s'ha anat explicant durant aquest capítol:

Segons aquest DTD, un missatge enviat al servidor és anomenat pel sistema `<MESSAGE>` i aquest pot contenir un o més logs `(LOG)+`.

Un element `<LOG>` correspon a una sessió sencera del context d'ús entre l'usuari i l'aplicació. Donat el cas que en una sessió no es pogués enviar l'XML amb aquesta informació sobre el context d'ús al component servidor, el component client ha de ser capaça d'enmagatzemar-lo i en una sessió posterior, continuar monitoritzant el context en el mateix fitxer XML en forma de un nou `<LOG>`.

Cal destacar també que cada un dels `<LOG>` enviats té dos elements importants: `<HEAD>` i `<BODY>`. El primer conté metainformació que és essencial per al component servidor per a poder identificar el component client que esta intentant comunicar-se amb ell a l'hora de rebre el

missatge. L'altre element, `<BODY>`, conté la informació monitoritzada durant la sessió com ara els temps entre pantalles, els widgets i les seves característiques, les fonts i els events.

- **Servidor-client**
  El component servidor és qui envia en format XML els canvis proposats al component client. Donat que el capítol següent s'estudia la capacitat d'adaptació del component client s'ha cregut més adient mostrar-hi allí el DTD corresponent per aquest cas.

8.4 Tecnologies provades i descartades

- **EclipseME**
  EclipseME és un plugin que permet desenvolupar projectes per a dispositius mòbils amb J2ME sota Eclipse. Aquest plugin va deixar de rebre suport i actualment ha estat substituït per MTJ. Requereix tindre WTK ó JavaME SDK instal·lats al sistema. Diversos errors a l'hora de construir el paquet per culpa d'una incompatibilitat de versions entre aquest i l'Eclipse.

- **JavaME SDK**
  És el substitut en l'actualitat de WTK, inclou llibreries noves i d'altres actualitzades. Ha estat descartat perquè actualment només és suportat en la plataforma Windows.

- **Major & Ferrari**
  Es tracta d'una llibreria que permet capturar crides a mètodes, atributs, instàncies, etc, semblant a les funcionalitats que ofereix AspectJ, però actúa a més baix nivell. Això li permet poder capturar crides que resulten invisibles per a AspectJ. Aquesta llibreria està basada en una tecnologia anomenada Ferrari, que es comunica amb la JVM i permet aquesta captura a baix nivell. Bàsicament Major és una llibreria que permet l'ús de les funcionalitats de Ferrari de forma relativament més còmoda per al desenvolupador.
  Es va pensar que es necessitaria d'aquesta funcionalitat per tal de poder capturar mètodes de la llibreria gràfica que es criden implícitament. Això és, per exemple, quan un mètode no es crida des d'on un aspecte d'AspectJ tindria accés, és a dir des de l'aplicació, sinó que la seva crida prové, per exemple, perquè es requereix redibuixar una finestra quan la finestra que tenia una capa per sobre s'ha mogut.
  Major va donar força problemes quant a configuració i ús, i a més

només servia per a J2SE (J2ME, afortunadament, és un entorn força més controlat on n'hi ha prou amb les captures de mètodes cridats explícitament). Finalment es va descartar ja que es van trobar solucions alternatives basades en AspectJ on no feia falta capturar crides implícites d'events.

- **Groovy & Grails**
  Aquesta recerca parteix d'una prèvia aplicació client servidor realitzada per un antic alumne: Jordi Monné. El servidor estava implementat en Grails. El client era un lector de 'Feeds' fet en J2ME amb la llibreria gràfica LCDUI. Abans de fer la migració a LWUIT es va tenir que entendre el funcionament d'aquest servidor. Llavors mencionar que Groovy és una alternativa al llenguatge Java i Grails és un framework que extèn el llenguatje Groovy.
  Finalment, per aquest projecte es va preferir l'opció J2EE + Tomcat gràcies a que és més entenedora, més utilitzada i per tant més documentada a la web i per la qual hi ha moltes més eines com podria ser GWT, també molt utilitzades.

- **AJDE**
  És un plugin que permet desenvolupar projectes amb AspectJ des de l'IDE Netbeans. Es va provar per intentar desenvolupar el projecte íntegrament amb Netbeans ja que aquest IDE integra força millor les tecnologies pel desenvolupament de J2ME però el suport d'AspectJ a Eclipse està força més madur. Aquest plugin funciona força be quan s'ha d'utilitzar amb J2SE però és incapaç de compilar un projecte J2ME amb suport per a AspectJ.

- **Jess**
  Jess és un motor d'inferència basat en regles i desenvolupat en Java. Aquest motor té dos inconvenients per a aquest projecte que han fet que s'acabés utilitzant Drools Expert en comptes d'ell. Aquest defectes són la seva llicència privativa (tot i que ja s'havia otorgat al projecte una llicència educativa permetent el seu ús), i el sobrecost que provoca haver de definir la base de coneixement en un llenguatge diferent del de Java abans de poder introduir les dades al motor d'inferència.

- **Jetty**
  Jetty és un servidor d'aplicacions que actualment és mantingut per la fundació Eclipse i és famós per ser molt lleuger. Es va descartar perquè era complicat configurar-lo correctament per al desplegament de l'aplicació web.

- **Swing Look And Feel** (LAF)
  Swing suporta l'ús de temes per la decoració dels elements de la interfície d'usuari. Aquests temes, o Look And Feel a l'argot de Swing, es van estudiar per tal d'aplicar un canvi notori a la interfície i després deixar que el motor d'inferència anés polint els detalls mitjançant la proposta de petits canvis o be directament proposar l'ús d'un nou LAF. Aquesta branca del projecte, però, va ser descartada tot i haver construït una demostració que era capaç de canviar el LAF d'una aplicació programàticament de forma dinàmica (és a dir, mentre l'aplicació estava sent utilitzada, descarregant el LAF via HTTP i sense tocar el codi de l'aplicació en cap moment).

  El principal motiu per descartar-la va ser el nivell extra de complexitat que calia afegir al protocol de comunicació entre component client i component servidor, i especialment l'augment de complexitat quant a la fabricació de regles pel motor d'inferència, ja que s'hauria de tindre també en compte el LAF aplicat a l'hora de suggerir canvis. A més a més es perdia una de les major virtuds de la solució, la abstracció de la llibreria gràfica emprada per l'aplicació, ja que aquesta tècnica només és compatible amb Swing.

- **SQLite**
  SQLite és un gestor de base de dades relacional de lliure distribució molt lleuger. Està escrit en C i ocupa força menys d'1Mb. No s'executa com a servidor sinó que s'utilitza en forma de llibreria. Actualment està integrat a molts grans projectes, que van des de Firefox i Skype a la majoria dels sistemes operatius per a mòbils(Android, iOS, Symbian, Maebo i Meego entre d'altres). Es va descartar SQLite perquè MySQL disposa de millors eines d'administració i ús i una comunitat molt més gran.

- **Spring AOP**
  Spring és un framework molt popular al món Web. Es tracta d'un framework amb un disseny modular que pretén simplificar i fer més robusta i eficient la programació de pàgines web amb Java.

  Una de les seves característiques és que té el seu propi framework AOP, el qual pot ser utilitzat sense sobrecarregar la aplicació amb parts innecesàries gràcies al disseny modular de Spring. De fet el propi framework utilitza la seva implementació d'AOP per implementar *cross-cutting*

concerns com seguretat, transaccions...

Comparat amb AspectJ és menys potent, més senzill d'utilitzar i possiblement requereix menys recursos.

Spring AOP està basat en intercepcions. Això és a l'hora el taló d'Aquiles (ja que limita l'actuació dels aspectes a mètodes *public ó* protected) en objectes existens i a l'hora un punt interessant ja que amb aquesta limitació ens estalviem el pas previ de compilació, és a dir, funciona com una llibreria normal, a diferència d'AspectJ, que requereix del procés de *weaving).*

Capítol 9

Capacitat d'adaptació de la solució proposada

9.1 Introducció

Una de les parts més curioses del projecte és l'adaptació de la interfície d'usuari. Actualment el component client és capaç d'adaptar la interfície d'usuari dinàmicament i automàticament, en qualsevol moment que li arribi un canvi. El sistema, però, ha estat limitat ja que aplicar canvis en qualsevol moment podria perjudicar l'ús de la Interfície d'Usuari (IU) ja que despistaria a l'usuari. És per això que la proposta d'adaptació es mostra únicament cada cop que s'inicia l'aplicació. S'ha vist en capítols anteriors que aquesta proposta d'adaptació es genera al component servidor en el motor d'inferència, mitjançant l'aplicació de regles prèviament definides pels administradors del sistema. Aquestes regles provoquen una proposta, formada per una sèrie de canvis a la interfície d'usuari. Aquest capítol es centra en explicar quins són els canvis que les regles poden proposar, així com quines limitacions i peculiaritats tenen.

9.2 Homogeneïtat vers les llibreries gràfiques

Aquest és potser el concepte més important del capítol. S'ha fet un esforç per a que la definició de canvis sobre la IU, fós completament independent del component client. Això es pot veure com un inconvenient ja que limita el ventall de possibles canvis als especificats pel protocol propi, que serà definit després. No obstant esdevé en un gran avantatge quant a la definició de regles, ja que, a banda de simplificar la seva escriptura, les regles poden ser reutilitzables. De fet una regla pot ser aplicada sense necessitat de

modificar-la ni el més mínim a qualsevol aplicació objectiu i client, independentment de si es tracta d'un client J2ME, J2SE, o algun altre soportat en un futur. Tampoc influeix, doncs, si l'aplicació està construída amb Swing, AWT, SWT, LCDUI, LWUIT, ó qualsevol altra llibreria que sigui suportada en un futur al component client.

9.3 Adaptacions permeses pel protocol

Actualment el protocol suporta els següents canvis a qualsevol element de la interfície gràfica:

- Vertical size
- Horizontal size
- Background color
- Foreground color
- Font size
- Font style
- Font type

El nombre de canvis suportats pot crèixer en un futur, simplement expandint el protocol. Cal advertir també que, de la llista de canvis possibles, només podran esdevenir efectius aquells que la llibreria gràfica suporti. Es recorda doncs que apareixen tres limitacions quant a aplicació de possibles canvis:

- La limitació que imposa el protocol, és a dir, que el canvi que es vol proposar no aparegui a la llista del paràgraf anterior
- La limitació de la llibreria gràfica, és a dir, que el canvi que es vol proposar no sigui suportat per la llibreria gràfica. Per saber més sobre aquestes limitacions consulti el capítol següent
- La limitació de l'UIAdapter, és a dir, la limitació de la implementació concreta per una llibreria gràfica de la classe que s'encarrega d'aplicar els canvis al component client. Fòra el cas d'un canvi que estigui especificat al protocol, sigui viable fer-lo amb la llibreria gràfica utilitzada

per l'aplicació, però el component client sigui incapaç de transmetre aquesta informació a la llibreria gràfica

9.4 Especificació del protocol de comunicació servidor-client

Els canvis són enviats via XML al component client. El XML ha de tindre el format especificat aquí per a que el component client el pugui parsejar i entendre correctament.

El document ha de tindre com a element arrel `<PROPOSAL_SET>`. Dins l'element arrel es poden trobar tantes propostes de canvi com es vulguin. Cada proposta representa un canvi lógic que s'aplicarà a la interfície d'usuari. Per exemple, es pot voler canviar el color de fons de tots els elements de tipus JLabel a blau, això representa un canvi lògic, que després podrà esdevenir en més d'un canvi físic a la IU, ja que pot haver més d'un JLabel, i tots canviaran el seu color de fons a blau. L'element XML que representa una proposta és el següent: `<PROPOSAL>`.
Una proposta ha de contenir exactament dos elements: `<CONSTRAINTS>` i `<CHANGES>`. El primer especificarà a quins elements de la interfície d'usuari s'aplicaran els canvis. Per tal de determinar exactament a quins `<CONSTRAINTS>` podrà tindre un o més elements `<WIDGET>`.
L'element `<WIDGET>` no representa un element de la interfície d'usuari sinò una regla per seleccionar un nombre d'ells. Així, si es té, per exemple, un element `<WIDGET>` amb l'element `<TYPE>` especificant JLabel, només els elements de la interfície que siguin del tipus JLabel rebran els canvis proposats en aquest `<PROPOSAL>`. Si s'especifiquen varies propietats dins l'element `<WIDGET>` els canvis només s'aplicaran als elements de la interfície d'usuari que cumpleixin TOTS els requisits. Si s'afegeix més d'un element `<WIDGET>` dins `<CONSTRAINTS>` els canvis s'aplicaran als elements de la IU que cumpleixin totes les propietats especificades a qualsevol dels `<WIDGET>`.

Simplificant, les propietats definides dins `<WIDGET>` actuen amb lògica AND mentre que el conjunt de `<WIDGET>` actua amb la lògica OR.

Suposem que es vol canviar el color de la lletra a blanc de tots els `<WIDGET>` que siguin del tipus `JLabel` i tinguin un color de fons blau. Dins l'element `<CONSTRAINTS>` hi aniria el següent:
`<WIDGET>`

```
<TYPE>JLabel</TYPE>
<BACKGROUND_COLOR>0,0,255</BACKGROUND_COLOR>
</WIDGET>
```

En canvi si es vulgués aplicar aquest canvi només als elements de la interfície que tinguessin color blau o bé que fóssin del tipus `JLabel` es definirien les restriccions de la següent forma:

```
<WIDGET>
<TYPE>JLabel</TYPE>
</WIDGET>
<WIDGET>
<BACKGROUND_COLOR>0,0,255</BACKGROUND_COLOR>
</WIDGET>
```

D'aquesta manera els canvis s'aplicarien a molts més elements que no pas en l'exemple anterior, ja que no caldria que les dues restriccions fóssin certes per a cada widget a l'hora.

Finalment, el element del document on estan especificats els canvis és `<CHANGES>`. Aquest element pot contenir només un element `<WIDGET>`, les propietats del qual definiran els canvis a realitzar.

I quines propietats permet l'element `<WIDGET>`? Es poden utilitzar els següents elements dins el tag `<WIDGET>`:

- `<VSIZE>` Representa el tamany vertical d'un element
- `<HSIZE>` Representa el tamany horitzontal d'un element
- `<BACKGROUND_COLOR>` Representa el color de fons d'un element
- `<FOREGROUND_COLOR>` Representa el color en primer pla d'un element (per exemple, el color del que es renderitzaran les lletres, etc)
- `<FONT>` Representa una font de lletra, la qual pot tindre qualsevol de les propietats definides al següent paràgraf

L'element `<FONT>` pot tenir qualsevol dels elements següents:

- `<SIZE>` Representa el tamany de la font
- `<STYLE>` Representa l'estil de la font. Els estils reconeguts actualment són:

  - BOLD (Negreta)
  - ITALIC (Cursiva)
  - BOLDANDITALIC (Negreta i cursiva a l'hora)
  - NONE (Estil normal, ni negreta ni cursiva)

- `<TYPE>` Representa la familia de la font, que pot ser qualsevol de les famílies disponibles al sistema client

Cal remarcar que els colors han de ser especificats a l'estil r,g,b (RGB, Red Green Blue), amb una precisió de 8 bits per canal. Això dóna una profunditat de color total de 24 bits, i el valor de cada canal va de 0 a 255, éssent 0 la intensitat mínima d'un canal i 255 la intensitat màxima permesa. Per qualsevol dubte en el format en que s'han d'especificar les dades es recomana consultar els logs generats pel component client, ja que el format d'entrada del component client és similar al format de sortida. Per qualsevol altre dubte en la formació de les propostes consultar el DTD (Document Type Definition, un document on s'especifica un cas concret del meta-lleguatge XML) proporcionat al final del capítol.

Quant a la família de la font, s'ha d'aclarir que obviament la font (com a recurs) no és proporcionada per la proposta de canvi, sinò només el nom o referència a la la mateixa. Això implica que es pot proposar un canvi especificant una font que el client no tingui. Si és aquest el cas el canvi no es pot aplicar i el la font serà renderitzada com si no s'hagués especificat un canvi de familia. Per saber quines fonts estan permeses a un sistema concret i quina format ha de tindre la seva referència hom pot consultar els logs proporcionats pel component client.

Tot completant aquesta informació i com s'ha mencionat en el capítol anterior a continuació es mostra aquest format de comunicació en la seva versió DTD:

```
 <!DOCTYPE PROPOSAL_SET [
<!ELEMENT PROPOSAL_SET (PROPOSAL+)>
        <!ELEMENT PROPOSAL (CONSTRAINTS,CHANGES)>
                <!ELEMENT CONSTRAINTS (WIDGET*)>

<!ELEMENT WIDGET (HSIZE, VSIZE, FOREGROUND_COLOR, BACKGROUND_COLOR, TYPE, FONT)>
                <!ATTLIST WIDGET ID ID #REQUIRED>
<!ELEMENT HSIZE (#PCDATA)>
<!ELEMENT VSIZE (#PCDATA)>
<!ELEMENT FOREGROUND_COLOR (#PCDATA)>
<!ELEMENT BACKGROUND_COLOR (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT FONT (TYPE, SIZE, STYLE)>
<!ELEMENT TYPE (#PCDATA)>
```

```
<!ELEMENT SIZE (#PCDATA)>
<!ELEMENT STYLE (#PCDATA)>

                    <!ELEMENT CHANGES (WIDGET)>

<!ELEMENT WIDGET (HSIZE, VSIZE, FOREGROUND_COLOR, BACKGROUND_COLOR, TYPE, FONT)>
                    <!ATTLIST WIDGET ID ID #REQUIRED>
<!ELEMENT HSIZE (#PCDATA)>
<!ELEMENT VSIZE (#PCDATA)>
<!ELEMENT FOREGROUND_COLOR (#PCDATA)>
<!ELEMENT BACKGROUND_COLOR (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT FONT (TYPE, SIZE, STYLE)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT SIZE (#PCDATA)>
<!ELEMENT STYLE (#PCDATA)>
```

Capítol 10

Casos d'estudi

En aquest capítol es volen demostrar dos casos d'estudi. Mitjançant aquests és vol demostrar al lector que el sistema és capaç de complir els seus objectius establerts.

10.1 Cas d'estudi A: Adaptació en LWUIT

En aquest cas d'estudi es vol fer una adaptació per un determinat contetx d'usuari que es disposa a utilitzar una aplicació per a un dispositiu mòbil.

10.1.1 Requeriments

Partirem d'una conjunt de requisits. Primer establirem els requisits que pertanyen al contexte d'usuari del discapacitat:

- Usuari amb discapactiat física
- Temblor
- Braç esquerra
- Dispositiu tàctil
- 50% d'afectació

A més a més es necessita concretar les capacitats del dispositiu mòbil:

- És capaç d'executar aplicacions J2ME

- Existeix definit al sistema en la base de dades de WURFL
- El dispositiu disposa de connexió WiFi

I també de l'aplicació per la qual es volen oferir propostes de canvis:

- Es té el codi font de l'aplicació
- Es sap que la llibreria gràfica que utilitza l'aplicació és LWUIT.

10.1.2 Objectius

Partint de totes les condicions establertes en l'anterior apartat i d'un estudi exhaustiu dels requisits d'aquest usuari discapacitat, es dessitja aconseguir el següent resultat que es creu que millorarà l'experiència de l'usuari:

- Tots els elements del tipus Button de la IU que són adaptables han de tenir un tamany d'un 20% per a contrarestrar la pèrdua de mobilitat de l'usuari.
- Tots els elements del tipus Button de la IU que són adaptables s'han de mostrar sobre un fons de color . Així també seran més visibles per l'usuari discapacitat.

10.1.3 Regles per al motor d'inferencia

El supervisor de l'usuari discapacitat, qui és l'expert, crea la següet regla per a que el motor d'inferència retorni els objectius dessitjats el punt anterior:

```
import cat.eps.tfc.server.web.server.beans.*;

global ProposalSet output

rule decideAdaptations
    when
eval(WidgetBean(type="Button"))
    then
UIRule rule = new UIRule();
WidgetBean wb = new WidgetBean();
wb.setVerticalSize(50);
wb.setHorizontalSize(50);
rule.setChanges(wb);
output.getRules().add(rule);

end

rule decideAdaptations2
    when
```

```
eval(true)
    then
UIRule rule = new UIRule();
WidgetBean wb = new WidgetBean();
wb.setBackgroundColor("0,0,255");
rule.setChanges(wb);
output.getRules().add(rule);

end
```

El resultat és mostra en la següent comparativa d'imatges:

Figura 10.1: Abans i després per a LWUIT

10.2 Cas d'estudi B: Adaptació en SWING

En aquest cas d'estudi es vol fer una adaptació per un determinat contetx d'usuari que es disposa a utilitzar una aplicació per a un ordinador de sobretaula.

10.2.1 Requeriments

Partirem d'una conjunt de requisits. Primer establirem els requisits que pertanyen al contexte d'usuari del discapacitat:

- Entorn en un ordinador de sobretaula.
- Usuari amb discapactiat visual

- Miopia
- Ambdós ulls
- 50% d'afectació

A més a més es necessita concretar les capacitats del PC:

- Disposa d'una màquina virtual de Java

I també de l'aplicació per la qual es volen oferir propostes de canvis:

- Es té el codi font de l'aplicació
- Es sap que la llibreria gràfica que utilitza l'aplicació és SWING.

10.2.2 Objectius

Partint de totes les condicions establertes en l'anterior apartat i d'un estudi exhaustiu dels requisits d'aquest usuari discapacitat, es dessitja aconseguir el següent resultat que es creu que millorarà l'experiència de l'usuari:

- Tots els elements de la IU que són adaptables han de veure's augmentats en un 50% per contrarestrar la pèrdua de visió de l'usuari
- Tots els elements de la IU que són adaptables s'han de mostrar sobre un fons de color verd

10.2.3 Regles per al motor d'inferencia

El supervisor de l'usuari discapacitat, qui és l'expert, crea la següet regla per a que el motor d'inferència retorni els objectius dessitjats el punt anterior:

```
import cat.eps.tfc.server.web.server.beans.*;

global ProposalSet output

rule decideAdaptations
    when
eval(true)
    then
UIRule rule = new UIRule();
WidgetBean wb = new WidgetBean();
wb.setBackgroundColor("0,255,255");
wb.setVerticalSize(50);
wb.setHorizontalSize(50);
```

```
rule.setChanges(wb);
output.getRules().add(rule);

end
```

El resultat és mostra en la següent comparativa d'imatges:

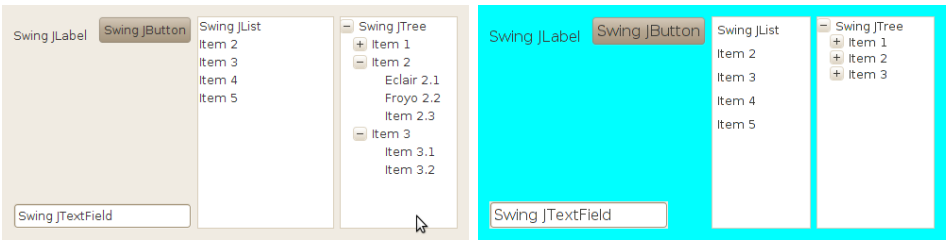

Figura 10.2: Abans i després per a SWING

10.3 Cas d'estudi C: Adaptació en SWT

En aquest cas d'estudi es vol fer una adaptació per un determinat contetx d'usuari que es disposa a utilitzar una aplicació per a un dispositiu sobretaula.

10.3.1 Requeriments

Partirem d'una conjunt de requisits. Primer establirem els requisits que pertanyen al contexte d'usuari del discapacitat:

- Entorn en un ordinador de sobretaula.
- Usuari amb discapactiat visual
- Miopia
- Ambdós ulls
- 50% d'afectació

A més a més es necessita concretar les capacitats del PC:

- Disposa d'una màquina virtual de Java.

I també de l'aplicació per la qual es volen oferir propostes de canvis:

- Es té el codi font de l'aplicació
- Es sap que la llibreria gràfica que utilitza l'aplicació és SWT.

10.3.2 Objectius

Partint de totes les condicions establertes en l'anterior apartat i d'un estudi exhaustiu dels requisits d'aquest usuari discapacitat, es dessitja aconseguir el següent resultat que es creu que millorarà l'experiència de l'usuari:

- Tots els elements de la IU que són adaptables han de veure's augmentats en un 50% per contrarestrar la pèrdua de visió de l'usuari.
- Tots els elements de la IU que són adaptables s'han de mostrar sobre un fons de color verd.

10.3.3 Regles per al motor d'inferencia

El supervisor de l'usuari discapacitat, qui és l'expert, crea la següet regla per a que el motor d'inferència retorni els objectius dessitjats el punt anterior:

```
import cat.eps.tfc.server.web.server.beans.*;

global ProposalSet output

rule decideAdaptations
    when
eval(true)
    then
UIRule rule = new UIRule();
WidgetBean wb = new WidgetBean();
wb.setBackgroundColor("0,255,255");
wb.getFont().setSize(16);
rule.setChanges(wb);
output.getRules().add(rule);

end
```

El resultat és mostra en la següent comparativa d'imatges:

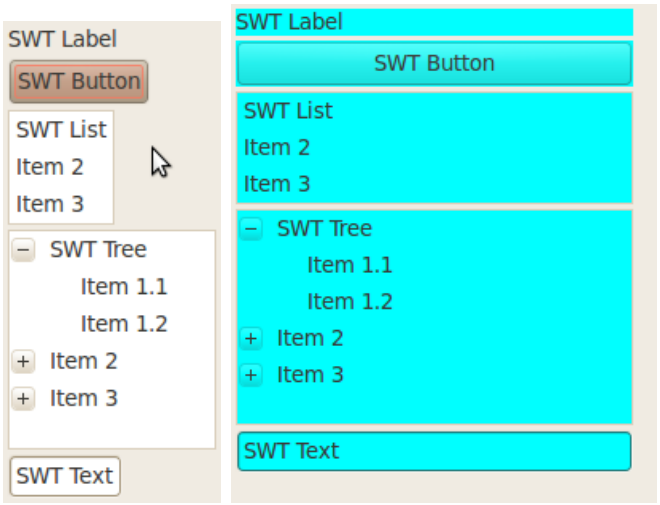

Figura 10.3: Abans i després per a SWT

Capítol 11

Conclusions i treball futur

11.1 Conclusions

Els objectius d'aquest projecte han anat creixent durant el seu desenvolupament. Inicialment es pretenia adaptar dinàmicament únicament un lector de feeds per a dispositius mòbils (referència al treball de'n Jordi Monné[6]).

De seguida, però, es va proposar d'ampliar aquest horitzó. Gràcies a la inspiració que va suposar per a nosaltres el descobriment del paradigma de la programació orientada a aspectes, sorgí la idea de dur a terme el sistema pensat un pas més enllà i intentar oferir una solució genèrica, que no es limités només a adaptar una aplicació específica. Ja que es feia una solució genèrica també es va decidir intentar afegir a la equació l'entorn d'escriptori.

Més tard, un cop es va veure que una solució d'aquest tipus era viable va ser el torn de'n, de vegades estimat de vegades odiat, Juan Miguel López, codirector del projecte. Ell va proposar anar una altre pas més enllà amb la idea d'automatitzar el procés d'injecció del component client a les aplicacions. Aquesta idea acabaria esdevenint el bloc web.

Hom podria pensar que ha sigut fàcil, doncs, el desenvolupament d'aquest projecte, ja que s'han ampliat les premises reiteradament. No obstant, res més lluny de la realitat. El camí ha sigut llarg, molt llarg, i també tortuós. Ens hem hagut d'avarallar, literalment, amb entorns de desenvolupament madurs, però que a l'hora d'integrar-los per a fer-los anar conjuntament resulten estar carregats d'incompatibilitats, poca documentació al respecte de molts temes tractats. Inclús al món web, una miríada de tecnologies cadascuna amb les seves restriccions, on algunes de les restriccions, a més, depenen

de la plataforma, i un llarg, molt llarg, etcètera.

No obstant, creiem que ha sigut un final feliç, i hem pogut demostrar que una solució d'aquestes característiques és viable. El resultat ha sigut una solució força madura i pulida, i amb una capacitat de ser extesa molt gran, ja que durant tot el desenvolupament del projecte s'ha tingut sempre en compte aquesta posibilitat d'ampliació a pràcticament tots els nivells. A més la solució es pot emprar per d'altres usos no pensats inicialment, aquests usos van des de l'adaptació dinàmica de la interfície d'usuari depenent de nous factors no proposats a aquest projecte fins a la posibilitat d'utilitzar-lo per a afegir noves funcionalitats a les aplicacions. Sense aner més lluny, bona part de la solució ha sigut emprada per una beca d'introducció a la recerca[8] en la que s'utilitzà per a incorporar estratègies de consciència de grup a una eina col·laborativa, com a mecanisme per a acomodar la interfície d'usuari al context d'ús i la capacitat cognitiva dels usuaris.

Moltes altres utilitats per aquest sistema estan esperant a ser descobertes.

11.2 Treball futur

Durant el desenvolupament d'aquest projecte per temes de recursos de temps i/o dificultat s'ha anat apuntant en quines parts es podria millorar tot el sistema, les llistem a continuació.

Afegir identificadors als widgets a l'afegir el component client a una aplicació
A l'hora de generar l'aplicació amb el component client incorporat, aquesta podria passar per un filtre per a detectar tots els widgets que apareixen al codi font que especifica la seva interfície d'usuari per a la llibreria gràfica en concret. Un cop localitzats tots els widgets, es podria injectar codi al codi font de l'aplicació de manera que aquests fossin identificables pel component client i reportats als logs. Així, mitjançant més regles al motor d'inferència al servidor, aquest seria capaç d'enviar propostes per modificar widgets especificant a quin en concret es volen aplicar els canvis, cosa que ara mateix no és possible realitzar ja que actualment només es poden proposar a un conjunt de widgets identificats per les seves propietats, i no per una ID única. No fa falta dir que el component client hauria de ser capaç, mitjançant els aspectes, de localitzar també aquests widgets i aplicar els canvis sobre ells. El protocol de comunicació també hauria de ser ampliat permetent identificar la ID de cada widget loguejat en forma de xml.

Tipus d'usuari en sessió
Tant el disseny de base de dades com el seu model, així com part del codi del servidor web esta preparat per suportar múltiples tipus de rol d'usuari en sessió. Actualment només només esta plenament funcionant l'usuari administrador. S'haurà d'acabar d'implementar en el servidor web aquesta funcionalitat.

Guardar propostes de canvis de forma que el motor d'inferència pugui utilitzar-los com a historial de resolucions
Actualment les propostes de canvi a la IU que genera el motor d'inferència són emmagatzemades a la base de dades en cru, en forma de XML, tal i com s'han enviat al component client. Fòra positiu per tal de poder escriure regles més precises i complexes que des d'una regla es pugués accedir a les propostes que s'han enviat prèviament al client i quin ha set el seu feedback, en forma de Beans. Per tal d'assolir això n'hi ha prou amb parsejar el XML de les propostes que s'han enviat al client i afegir-lo al conjunt de dades que manega el motor d'inferència, com a llista de Propostes de canvis.

Poder salvar l'estat de la creació de noves adaptacions
Actualment si es perd la connexió amb el servidor, l'adaptació falla per un motiu intern o simplement es desconnecta, tot el procés de creació d'adaptació s'ha de fer des de el principi. Una funcionalitat molt útil seria el poder guardar un llistat d'adaptacions que, per qualsevol motiu, no s'han pogut finalitzar correctament. Seleccionar-ne una i continuar des de l'últim pas.

Modificar adaptacions definides a la web
Actualment les adaptacions són immutables. Això implica que si es desitgen realitzar petits canvis en la seva definició l'única forma possible és modificant directament la base de dades. Si es vol canviar quelcom sense haver d'accedir a la base de dades s'ha d'esborrar la adaptació i tornar a crear-ne una des de zero.

Reports
Actualment la web permet filtrar i obtenir en forma de txt els logs de l'usuari, seria convenient posar-hi més condicions de cerca al filtre i utilitzar un servidor de reports open source com ara Jasper Reports per a poder visualitzar la informació de manera més organitzada.

Canals de comunicació
Actualment el pas de missatges entre component client i servidor només es

fa a través d'HTTP. Per a que les capacitats de comunicació fossin més variades es podria implementar la part receptora del servidor com a servidor Bluetooth així com enviar les dades des del client també via aquest canal. La solució actual contempla la ampliació de canals de comunicació de forma senzilla. Només cal implementar la interfície corresponent tal i com es mostra als annexes.

Actualització dinàmica de la base de dades de dispositius mòbils
Cada cop que ha de ser actualitzada la base de dades de perfils de dispositiu ha de ser feta a mà. Els intervals d'actualització són reduïts, el fitxer poc pesat i l'actualització es pot fer en calent, però la possibilitat de que aquest mecanisme es pogués fer des de la pròpia web de forma automàtica.

Desacoblar l'XML, abstraent el protocol de comunicació
L'XML està força arrelat a diferents llocs de tota la solució. Es possible abstreure el protocol de comunicació desacoblant l'XML dels Beans i les classes clau que l'utilitzin de forma que pugui ser substituït per altres protocols més eficients, segons el cas. Per exemple, si haguessin de conviure el component client i el component servidor a la mateixa màquina un protocol basat en POJO's serialitzats seria molt més eficient.

Component servidor incrustat
En comptes d'incrustar obligatòriament només el component client a l'aplicació, donar l'opció de, si el dispositiu objectiu té la suficient capacitat per manegar-ho, incrustar també el component servidor (o una versió reduïda del mateix). Aquesta opció es va contemplar al principi del projecte però de seguida va ser posposada per qüestions de homogeneïtat quant a l'estructura global del sistema, però amb una mica més d'esforç pot ser factible.

Guardar les propostes de canvi de forma estructurada
Actualment es manté a la base de dades un historial de les propostes de canvi que s'ha enviat al component client. El problema és que aquestes propostes s'emmagatzemen en cru, és a dir, es guarda íntegrament l'XML que s'envia al component client. Si es guardés a la base de dades de forma estructurada es podrien fer consultes sobre les propostes de canvi, i així podrien ser oferides també al motor d'inferència per tal de proporcionar més dades amb les que crear regles. Per fer viable aquesta opció caldria aplicar els següents canvis al projecte:

- Modificar la base de dades per a que els suporti

- Crear codi capaç d'estructurar i guardar aquesta informació al component servidor
- Crear codi que recuperi les propostes estructurades, les converteixi en POJO's i les injecti al motor d'inferència
- Documentar l'ús d'aquests nous objectes a la documentació de generació de regles del motor d'inferència, a la web

Suport per a aplicacions implementades amb altres llenguatges de programació diferents de Java
Actualment la solució proposada només suporta Java. El suport per a aplicacions implementades amb altres llenguantges, però, no resulta difícil gràcies a l'abstracció que ofereix el protocol de comunicació. De fet el sistema està preparat per a aquesta proposta. El que caldria fer, però, és implementar un nou component client en el llenguatge corresponent i afegir a la base de dades els arxius corresponents per a poder compilar les aplicacions.

Bibliografia

[1] Montserrat Sendín, *Infraestructura Software de Soporte al Desarrollo de Interfaces de Usuario Plásticas bajo una Visión Dicotómica*. 2007.

[2] Montserrat Sendín, Juan Miguel López , *Introducció al desenvolupament d'aplicacions per a dispositius mòbils*. Universitat d'Estiu UDL 2008.

[3] Juan Miguel López , *Estudio de las discapacidades visuales más comunas*. MIPO UDL 2008.

[4] Juan Miguel López , *Accesibilidad y discapacidades motoras*. MIPO UDL 2008.

[5] Juan Miguel López , *Accesibilidad y discapacidades auditivas*. MIPO UDL 2008.

[6] Jordi Monné, *Desenvolupament d'un lector i un agregador de feeds per a dispositius mòbils i de les eines necessàries per a dotar-la de diversos aspectes de plasticitat*. 2007.

[7] Jordi Viladrich, *Projecte de final de carrera de Jordi Viladrich*. 2006.

[8] Montserrat Sendín i Juanjo Pardo, *Interfaces de usuario plásticas y colaborativas*. 2010.

[9] Ramnivas Laddad , *AspectJ in Action*. Ed. Manning 2003.

[10] Joseph D. Gradecki, Nicholas Lesiecki , *Mastering AspectJ: Aspect-Oriented Programming in Java*. Ed. Wiley 2003.

[11] Robert T.Cooper, Charlie E.Collins , *GWT in Practice*. Ed. Manning 2008.

[12] Trevor J.Young , *Using AspectJ to Build a Software Product Line for Mobile Devices*. University British Columbia 2005.

Apèndix A

Preparació de l'entorn per al desenvolupament sobre dispositius mòbils (J2ME)

Descarregar i instal·lar en el mateix ordre els següents aplicatius:

- WTK Sun wireless toolkit 2.5.2_01
- Eclipse SDK IDE 3.5.2
- MTJ Mobile Tools for Java 0.9.1
- AJDT AspectJ Development Tools 2.0.2

Altres consideracions a tenir en compte:

- Plataforma: Linux x86
- Java SDK: 1.6 update 16

A.1 Eclipse

Modificar el fitxer eclipse.ini del directori d'Eclipse per a poder dedicar més memòria a l'IDE:
512m -vmargs -Xms128m -Xmx512m

A.2 MTJ

Instal·lar MTJ via Eclipse updates.

Per a configurar-lo correctament, primer cal indicar-li quin kit de desenvolupament es farà anar. En el cas d'estudi, el WTK.

Seguidament, a la pestanya de Device Management, importar els emuladors necessaris.

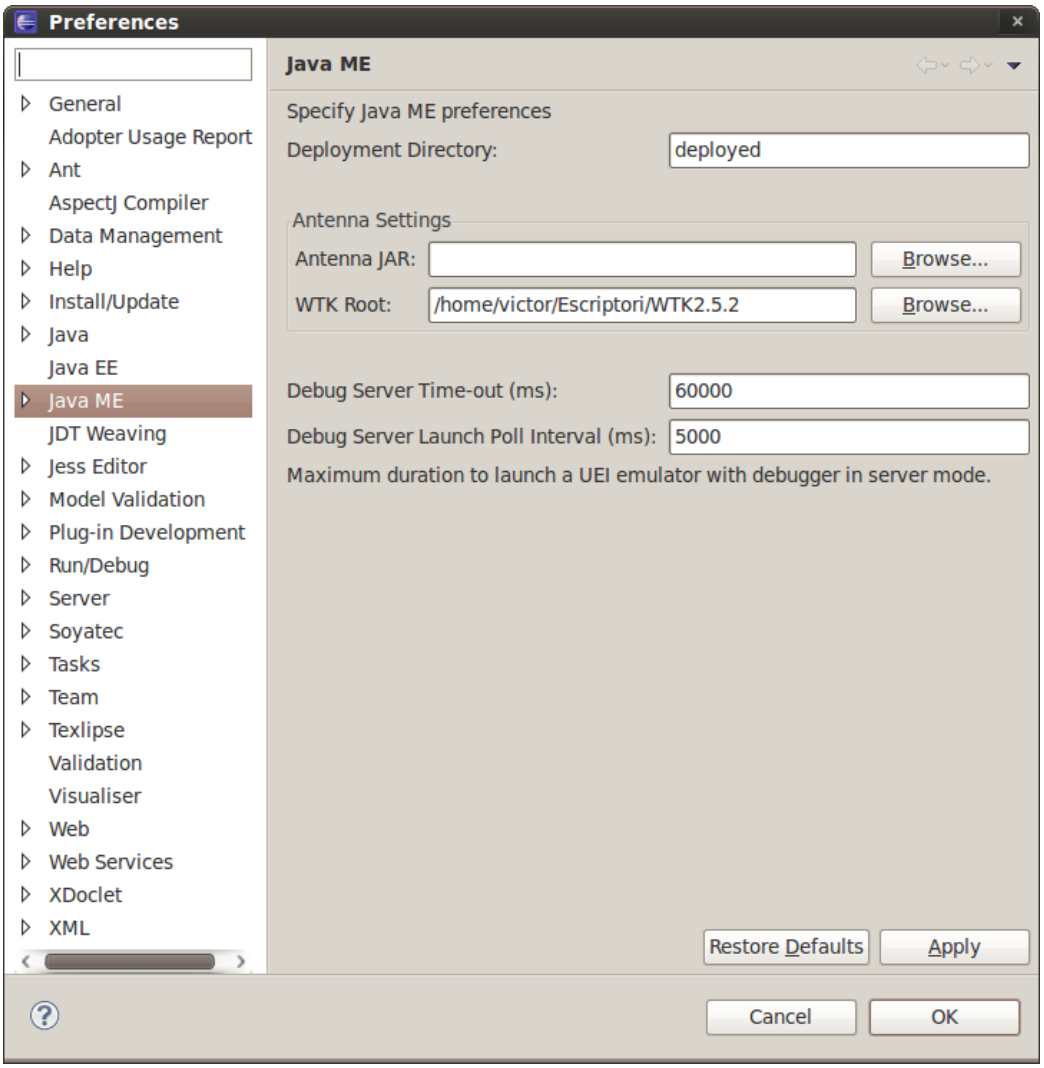

Figura A.1: Preparar l'IDE per a J2ME i WTK

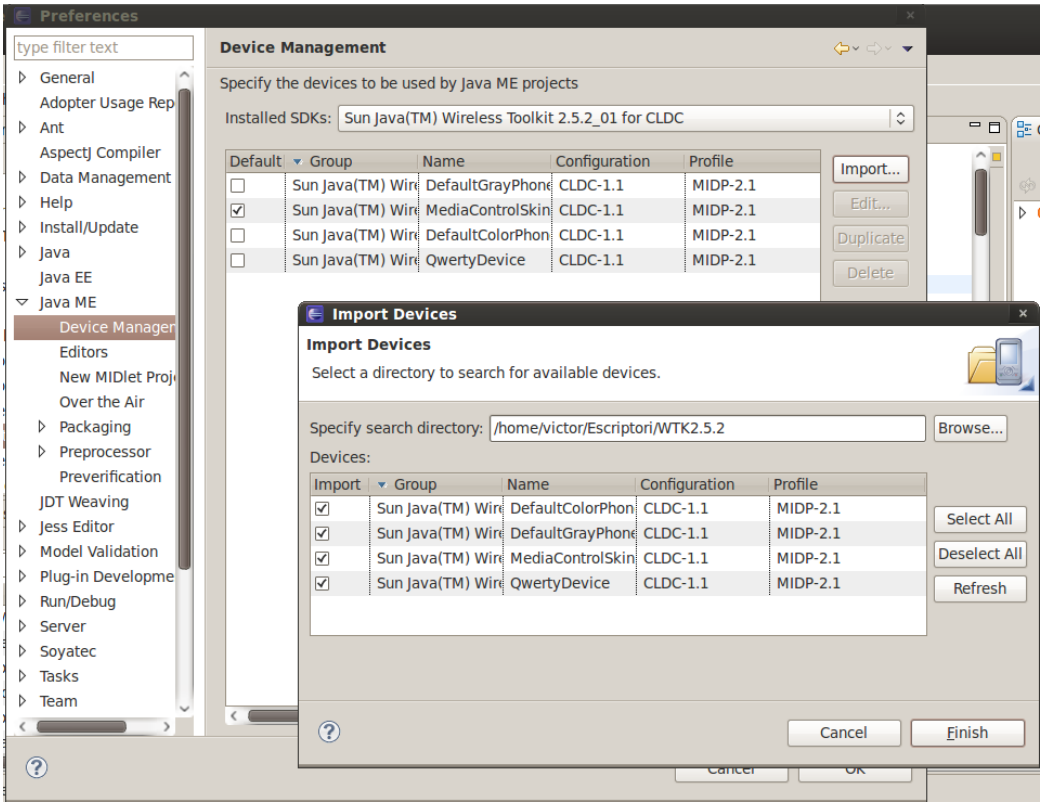

Figura A.2: Importar els devices que seràn emulats

A.3 AJDT

AJDT ha de ser instal·lat manualment. La versió que s'adjunta és la 2.0.0. Altres versions han set provades i també funcionen, únicament cal sincronitzar l'Eclipse amb el disc dur un cop fet el build (F5).

Es pot descarregar de: http://download.eclipse.org/tools/ajdt/35/update/ajdt_2.0.0_for_eclipse_3.5.zip

A la pestanya JDT Weaving seleccionar NO ACTIVAR el advanced weaving.

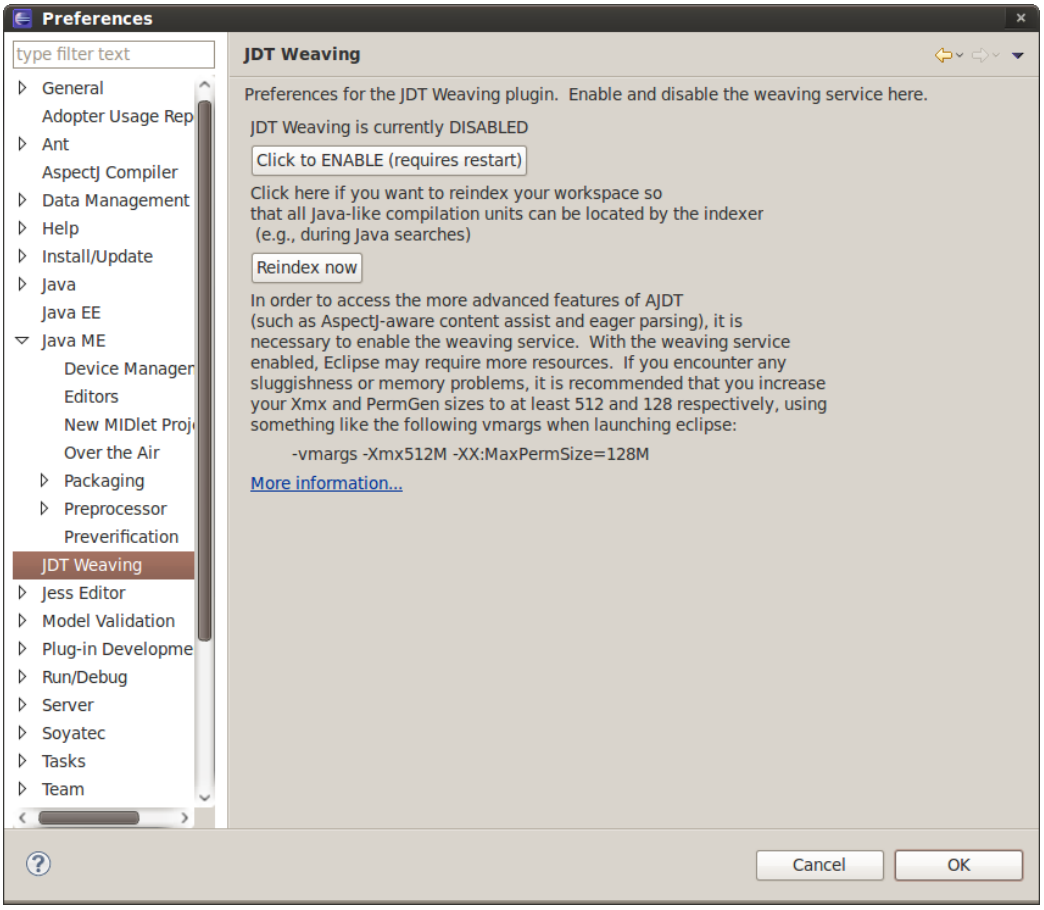

Figura A.3: Preferencies AJDT

A.4 Logger J2ME

Un cop configurat l'entorn, per a treballar amb la part J2ME del component client, s'ha d'importar el projecte a l'Eclipse.

Menu File → Import → Existing projects into workspace → Archive File → LoggerJ2ME.tar.gz.

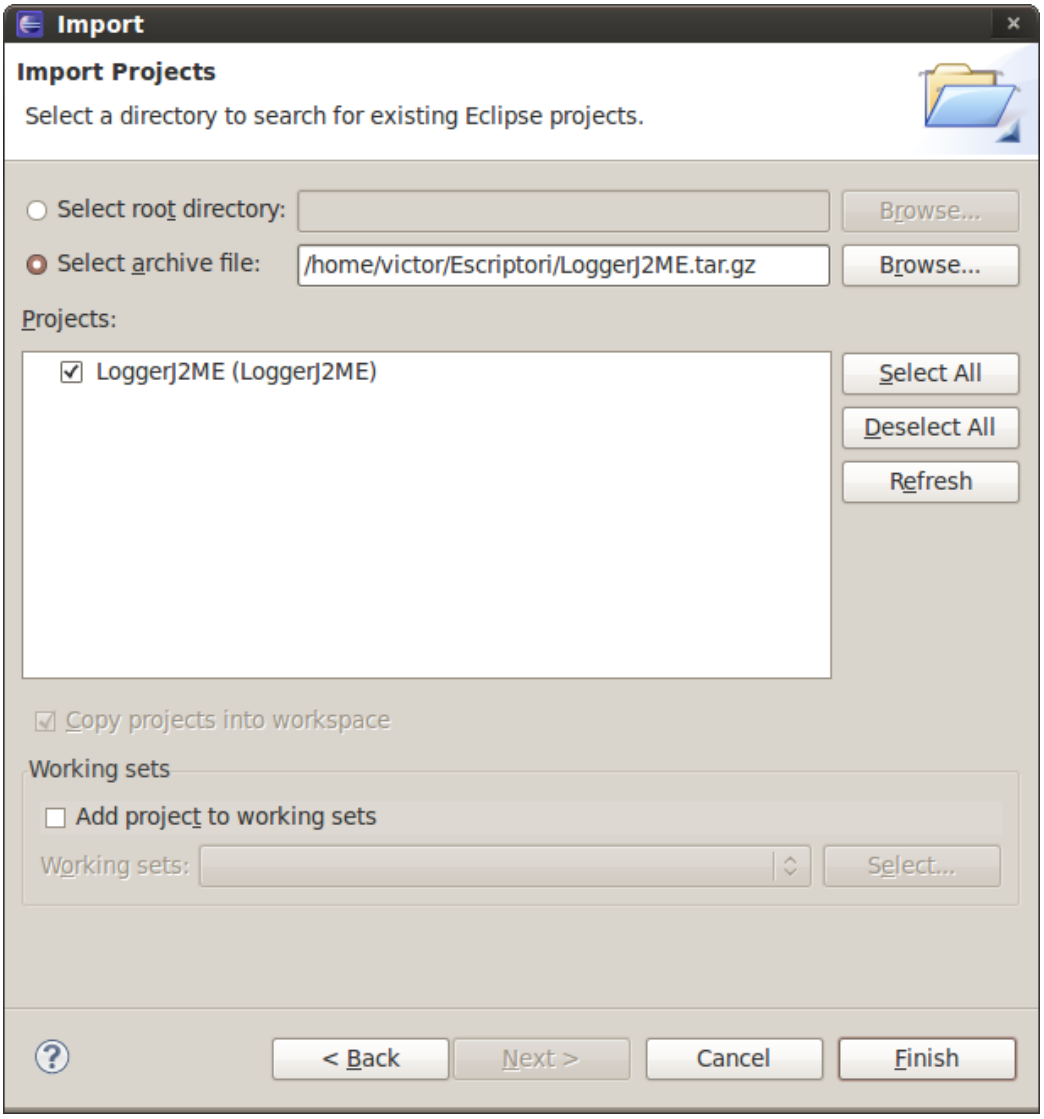

Figura A.4: Importar LoggerJ2ME a l'espai de treball

Apèndix B

Preparació de l'entorn per al desenvolupament d'escriptori (J2SE)

El component client per a J2SE s'ha desenvolupat sota la següent plataforma:

- GNU/Linux Ubuntu 9.10 per amd64
- Sun Java SDK 6

L'entorn pot funcionar fàcilment a altres plataformes com diferents distribucions de GNU/Linux i Microsoft Windows, així com sota arquitectures de 32 bits, tot i que aquest tutorial només contempla la opció especificada anteriorment.

B.1 SDK de Java

Per instal·lar el SDK de Java (més conegut com a JDK ó Java Development Kit) s'ha d'executar la següent comanda a una consola del sistema (com el gnome-terminal):

```
sudo apt-get install sun-java6-jdk
```

Per a altres plataformes (suportades) es pot obtindre el JDK de Sun (ara part de Oracle) des de la web de Oracle:
`http://www.oracle.com/technetwork/java/javase/downloads/index.html`

B.2 Eclipse

B.2.1 Instal·lació

S'utilitzà Eclipse com a entorn de desenvolupament. Així el primer pas és descarregar i instal·lar l'IDE Eclipse. Eclipse es pot descarregar des de:

```
http://www.eclipse.org/downloads/
```

Es recomana la versió "Eclipse IDE for Java Developers", però qualsevol que permeti desenvolupar en Java servirà. El número actual de versió d'Eclipse és el 3.6

Un cop descarregat l'arxiu n'hi ha prou amb descomprimir-lo. Durant tot el tutorial s'emprarà la ruta */home/entropy/eclipse/* com a ruta on s'ha descomprimit Eclipse.

B.2.2 Configuració

Workspace

A l'iniciar per primer cop Eclipse aquest ens demanarà la ruta del Workspace que volem utilitzar. El Workspace és la ruta on es desaran tots els projectes d'Eclipse. Durant tot el tutorial s'emprarà la ruta */home/entropy/workspace/* com a ruta del Workspace.

Registrar JDK

Un cop l'IDE s'ha iniciat s'ha de registrar la JDK del sistema dins Eclipse. Per fer-ho s'han de seguir els següents pasos:

1. Obrir la finestra de preferències, situada al menú, a *Window* → *Preferences*

2. Utilitzar l'abre de l'esquerra per anar a la pantalla de Installed JREs tal i com es mostra a la imatge

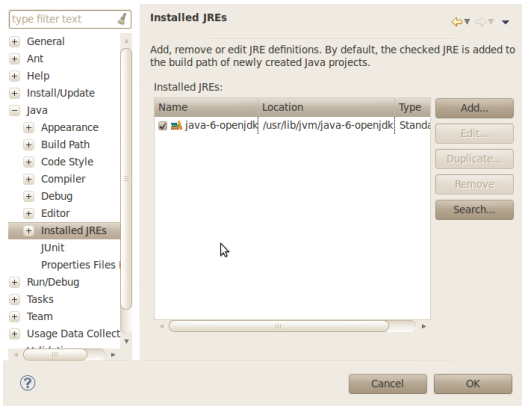

Figura B.1: JRE's instal·lades a Eclipse

3. Fer clic a *Add*

4. Seleccionar Standard VM, tal i com es mostra a la imatge

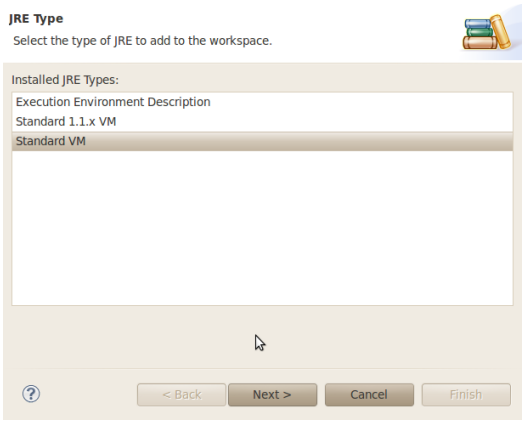

Figura B.2: Afegir una JRE a Eclipse

5. Especificar JRE home amb la ruta on s'ha instal·lat la JDK. A Ubuntu i la majoria de distribucions GNU/Linux aquesta ruta és */usr/lib/jvm/java-6-sun/*. Tal i com es mostra a la imatge, el camp del nom s'omple automàticament si la JDK s'ha detectat correctament.

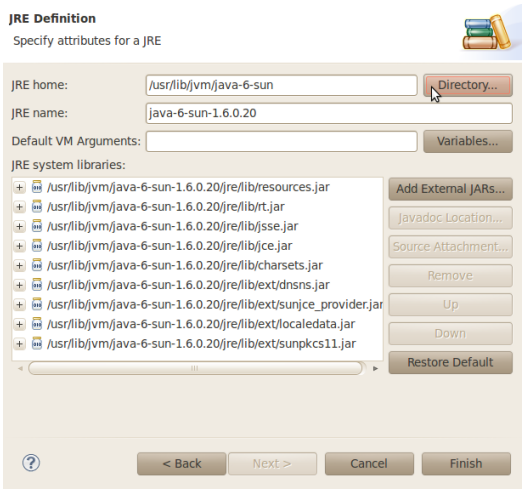

Figura B.3: Configuració dels paràmetres de la JRE

6. Fer clic a *Finish*

7. A la finestra de Installed JREs, a la llista ara apareixerà la nova JDK registrada. Marcar el checkbox que apareix abans del nom. Aquest checkbox indica quina és la VM que s'emprarà per defecte a Eclipse

8. Fer clic a *OK*

B.3 AspectJ

El component client utilitza AOP, en concret depen de la tecnología AspectJ. Per tal de poder desenvolupar amb aquesta tecnologia és necesari instal·lar un plugin a Eclipse. El plug-in d'Eclipse que ofereix aquesta funcionalitat és AJDT.

No s'han trobat problemes de compatibilitat en diferents versions d'AspectJ o AJDT i el desenvolupament dels components client basats en J2SE. Així en principi qualsevol versió d'Eclipse amb el plug-in corresponent, capaç de compilar aspectes serà funcional.

Hi ha diversos mètodes per instal·lar plug-ins a Eclipse. El més cómode i recomanable és utilitzar el Update Site. Per tal d'instal·lar el plug-in mitjançant l'Update Site cal seguir els pasos següents:

1. Obrir la finestra d'instal·lació de nou software, situada al menú, a *Help* → *Install New Software* Apareixerà la finestra mostrada a la imatge

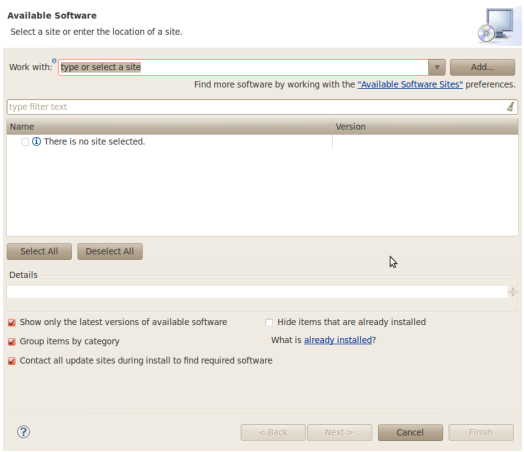

Figura B.4: Finestra de gestió de plug-ins d'Eclipse

2. Fer clic a *Add*

3. Omplir el formulari amb la URL de l'*Update Site*, tal com es mostra a la imatge. El nom és irrellevant

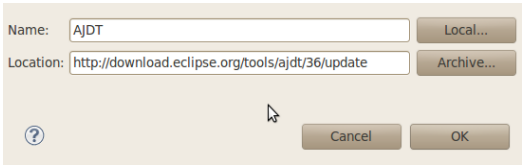

Figura B.5: Configuració d'un nou repositori de plug-ins

4. Fer clic a *OK*

5. Apareixerà la següent imatge, fer clic a *Next*

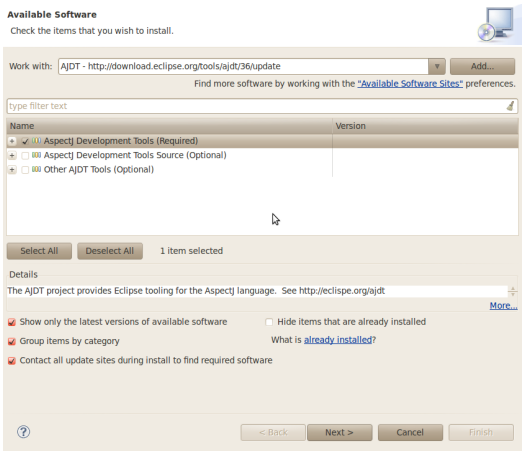

Figura B.6: Selecció de plug-ins a instal·lar

6. Apareixerà la següent imatge, fer clic a *Next*

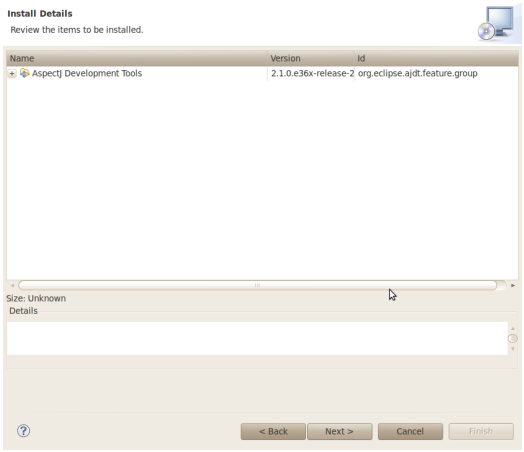

Figura B.7: Detalls de la instal·lació

7. Apareixerà la següent imatge, acceptar els termes de llicència i fer clic a *Finish*

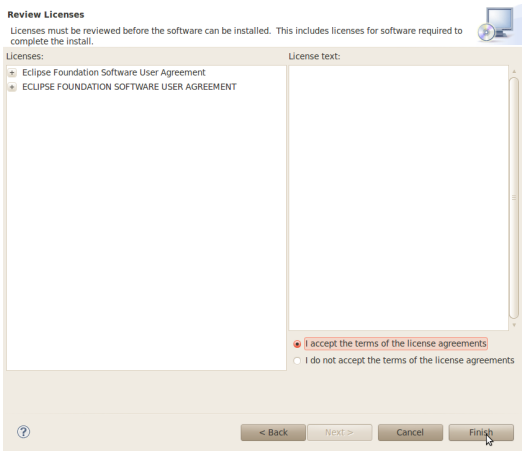

Figura B.8: Acceptació de les llicències dels plug-ins a instal·lar

8. Un cop Eclipse ha descarregat els arxius necesaris surtirà el següent avís, fer clic a *OK*

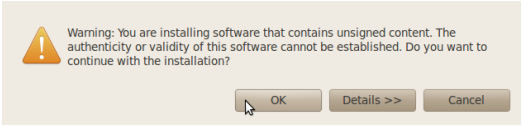

Figura B.9: Avís d'Eclipse

9. Un cop tot s'ha instal·lat correctarment apareixerà el següent missatge, informant que tot ha anat be i és necesari reiniciar Eclipse per a que el software instal·lat funcioni correctament. Fer clic a *Restart Now*

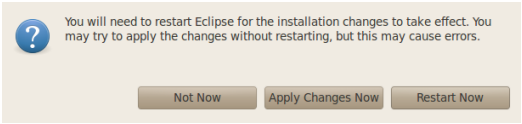

Figura B.10: Missatge d'Eclipse un cop instal·lats nous plug-ins

B.4 Comprovar que l'entorn funciona

Per comprovar que l'entorn ha estat instal·lat i configurat correctament es proposa crear un projecte a l'estil "hello world":

B.4.1 Creació del projecte

1. Al menú, anar a File → New → rightarrow Project...
2. A la finestra que s'ha obert seleccionar AspectJ Project (està dins la categoría AspectJ) i fer clic a Next
3. Inserir un nom pel projecte, com per exemple "Hello Aspect World"
4. Fer clic a Finish (ja que les opcions per defecte ens estan be)

B.5 Creació de la clase inicial

1. Al menú, anar a File → New → Class
2. Inserir "HelloWorldClass" com a nom de la clase i marcar l'opció d'autogenerar un mètode d'inici (public static void main), tal i com es mostra a la figura següent

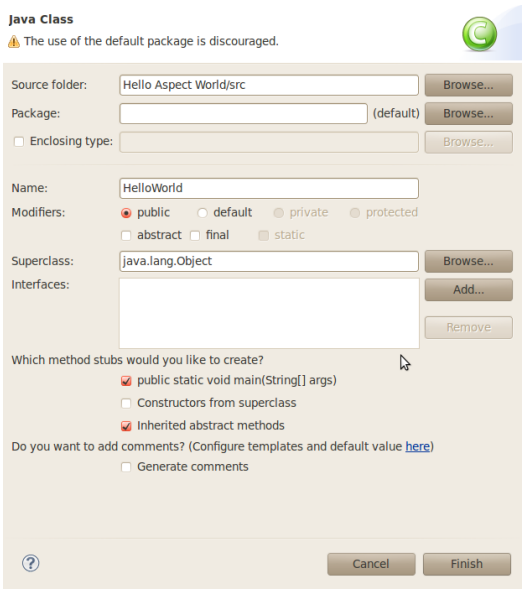

Figura B.11: Assistent per crear una nova classe

3. Fer clic a Finish
4. Al codi de la classe que acabem de crear, afegir la següent línia de codi dins el mètode main System.out.println("Hello World");

5. Desar els canvis a la clase (mitjançant la combinació de tecles control+s, per exemple)

B.6 Creació de l'aspecte

1. Al menú, anar a File → New → Other...
2. A la finestra que s'ha obert seleccionar Aspect (està dins la categoría AspectJ) i fer clic a Next
3. Inserir "HelloWorldAspect" com a nom de l'aspecte
4. Fer clic a Finish
5. Afegir el següent codi dins l'aspecte:

```
public pointcut mainPointcut() : execution(* main(*));
        before(): mainPointcut(){
                System.out.println("Hello Aspect World");
        }
```

6. Desar els canvis a l'aspecte (mitjançant la combinació de tecles control + S, per exemple)

B.7 Execució de l'exemple

1. Seleccionar el projecte a la vista Package Explorer d'Eclipse
2. Al menú, anar a Run → Run As... → AspectJ / Java Application
3. A la finestra que s'ha obert, seleccionar HelloWorldClass
4. Si tot ha anat be el programa s'executarà i imprimirà a la consola d'Eclipse el següent missatge:
   `Hello Aspect World Hello World`

Si no apareix la línia "Hello Aspect World" AspectJ no s'ha instal·lat / configurat correctament o s'han seguit incorrectament els pasos per crear el programa d'exemple

Apèndix C

Preparació de l'entorn per al desenvolupament del servidor

C.1 Aplicació Web

Baixar la versió Java de l'IDE Netbeans a:
`http://netbeans.org/downloads/`
i instalar-lo.

Descarregar i instal·lar l'apache tomcat de:
`http://tomcat.apache.org/download-60.cgi`
Més endavant configurar-lo com a servidor al Netbeans.

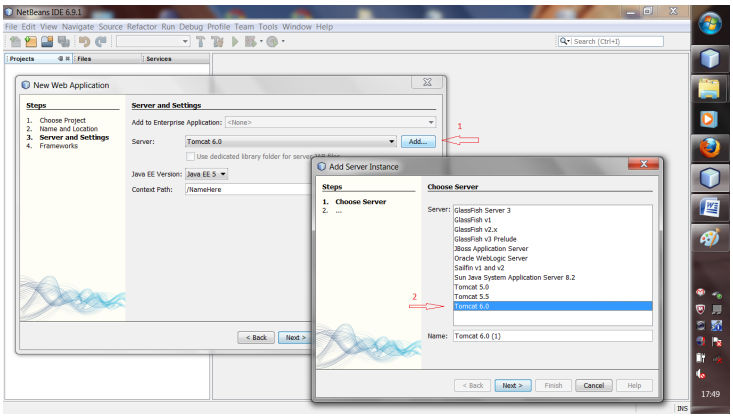

Figura C.1: Selecció de servidor a Netbeans

Baixar l'SDK de GWT:

`http://code.google.com/intl/es-ES/webtoolkit/download.html`
Descomprimir-lo al directori desitjat.

Des del gestor de plugins del Netbeans, descarregar el plugin de GWT4NB.

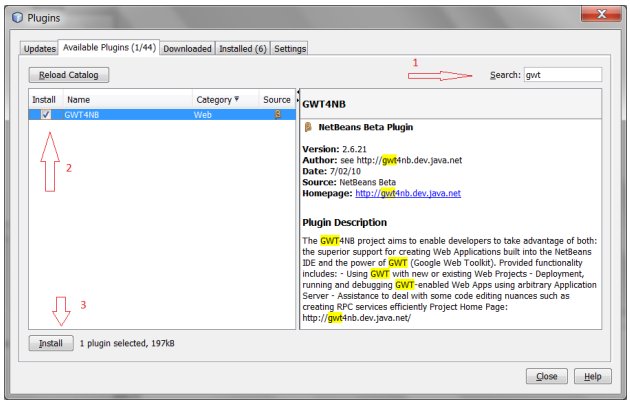

Figura C.2: Instal·lació del plug-in GWT4NB per a Netbeans

Copiar el contingut del codi del projecte a la carpeta de treball del Netbean. Després des de la vista de projectes fer click al botó dret per obrir-lo com a projecte i marcar-lo com a 'main project'. Caldrà reconfigurar si s'escau la ruta al framework de GWT (on s'ha descomprimit abans) des del menu del projecte a la pestanya frameworks.

Un cop obert s'han de resoldre les dependencies de llibreries, que es mencionen a continuació:

- Ant
  Directori lib de la distribució, que es pot descarregar de http://ant.apache.org/bindownload.
- Drools
  Directori lib de la distribució, que es pot descarregar de http://jboss.org/drools/downloads.h
- GWT
  Directori lib de la distribució del SDK descarregada anteriorment
- MySQL JDBC Driver
  http://www.mysql.com/downloads/connector/j/
- Commons
  http://commons.apache.org/

- Backport
  http://backport-jsr166.sourceforge.net/index.php
- JDom
  http://www.jdom.org/downloads/index.html
- WURFL
  http://sourceforge.net/projects/wurfl/files/WURFL

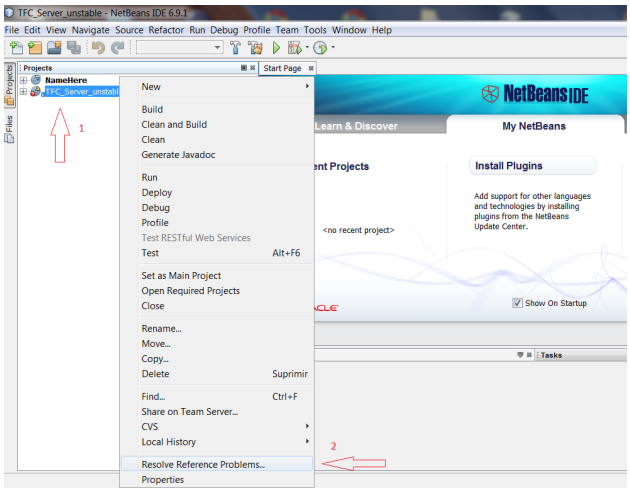

Figura C.3: Resoldre problemes de referència

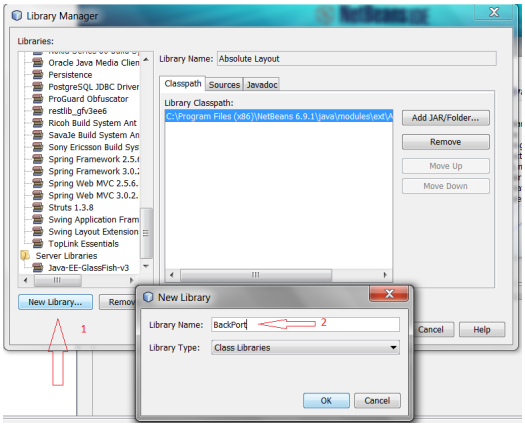

Figura C.4: Creació d'una llibreria a Netbeans

C.2 Servidor BBDD

El servidor de base de dades que s'ha utilitzat en aquest projecte és el MySQL en la versió no comercial 5.1.44. Aquest és lliure i gratuït, i es pot descarregar de la següent URL:
`http://www.mysql.com/downloads/`
Des d'aquí és recomanable descarregar-se la versió que porta incorporada una interficie d'usuari per administrar el servidor, anomenada MySQL Administrator i un client per llençar-hi sentencies SQL, MySQL Query Browser.

Un cop descarregat s'ha d'executar el programa d'instal·lació marcant que el servidor ha de ser instal·lat com a servidor dedicat. Un cop arrencat el Servidor haurem de crear un nou esquema, anomenat 'tfc'.

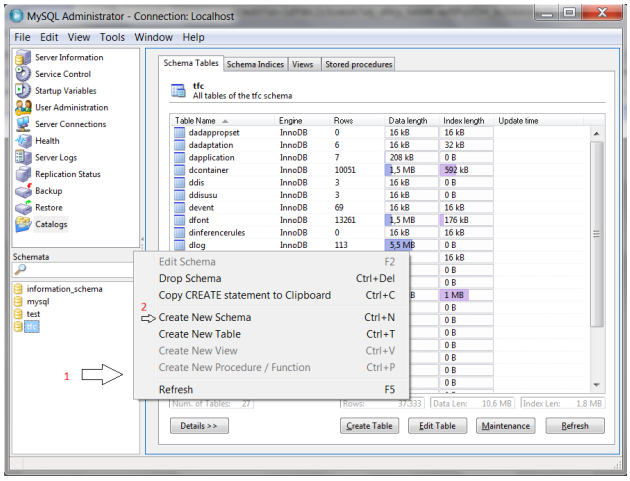

Figura C.5: Creació d'una nova base de dades

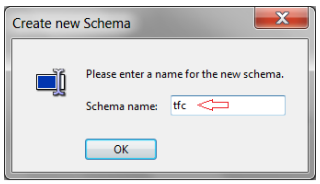

Figura C.6: Assistent per crear la base de dades

Per a poder restaurar totes les taules mestres s'ha entregat de forma digital un volcat de la BBDD. Un cop instal·lat el servidor haurem de fer un

"restore backup".

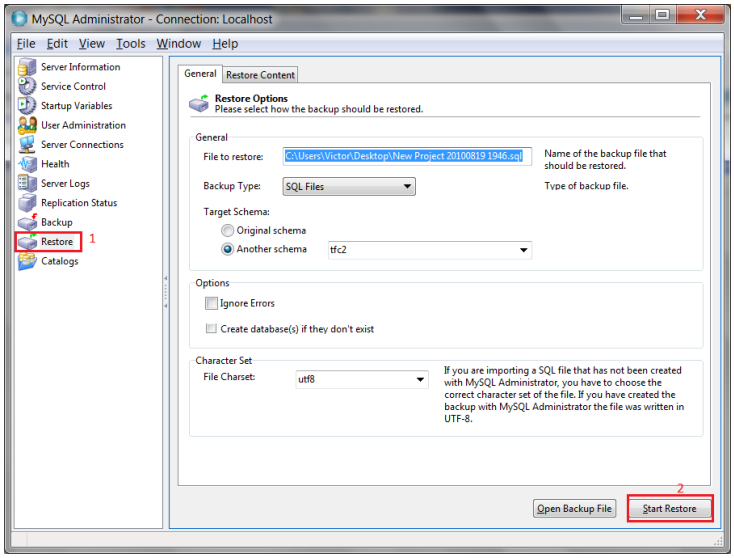

Figura C.7: Pantalla de restauració de la base de dades

Després de recuperar les dades haurem de llençar mitjançant el MySQL Query Browser la següent consulta per a poder guardar camps BLOB de tamany gran:

```
set global max_allowed_packet=1000000000;
```

Aquesta és tota la configuració necessària per a preparar el servidor.

És recomanat també l'instal·lació del client MySQL Workbench, també gratuït i molt més funcional que l'inclós en aquesta descàrrega.

Apèndix D

Model de base de dades

El model de dades complet es mostra en la figura següent:

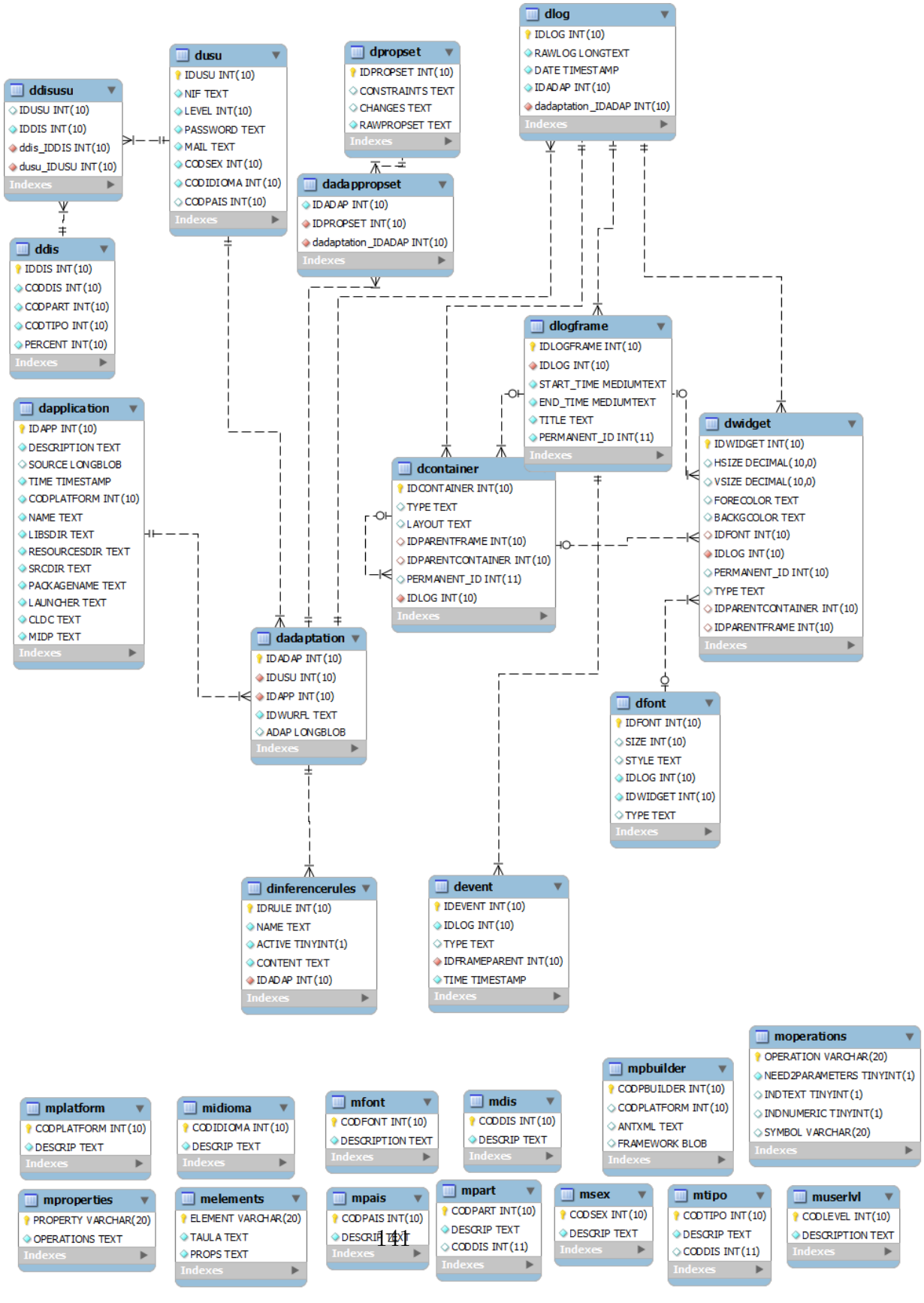

Figura D.1: Relacions entre taules

Les taules que comencen per la lletra 'M' son taules mestres que contenen informació mestra, és a dir, d'infraestructura. Són les taules que s'hauràn de modificar en cas de voler extendre les funcionalitats del sistema.

Les taules que comencen per la lletra 'D' son taules de dades 'volàtils'. Aquestes taules disposen totes d'indexos per tal de fer més efectiu l'access per al Sistema Gestor i per tant fer més usable l'aplicació.

En les pròximes seccions es detallen les més importants.

D.1 Definició de taules

DADAPTATION
Taula principal del model. Conté informació sobre l'adaptació.

- *IDADAP* Identificador únic de cada adaptació. ID principal del model
- *IDUSU* Identificador del perfil d'usuari
- *IDAPP* Identificador del perfil d'aplicació
- *IDWURFL* Correspon amb la ID del fitxer de WURFL per al dispositiu seleccionat.
- *ADAP* Conté l'aplicació amb el component client injectat

DDIS
Taula on es defineixen les discapacitats per a cada usuari. Està relacionada amb DUSU mitjançant DDISUSU.

- *IDDIS* Identificador de discapacitat
- *CODDIS* Codi de l'àmbit de discapacitat
- *CODPART* Codi de part afectada de la discapacitat
- *CODTIPO* Codi del tipus de discapacitat
- *PERCENT* Percentatge d'afectació

DUSU
Taula on es guarda l'informació de l'usuari.

- *IDUSU* Identificador de l'usuari
- *NIF* NIF de l'usuari

- *CODPAIS* Codi del pais de l'usuari
- *CODSEX* Codi del sexe de l'usuari
- *CODIDIOMA* Codi de l'idioma de l'usuari

DLOG
Taula on s'enmagatzemen els logs.

- *IDLOG* Identificador de log
- *RAWLOG* El log serialitzat
- *DATE* Data en la que s'ha creat el log
- *IDADAP* Clau forànea a IDADAP de DADAP

DINFERENCERULES
Taula on s'enmagatzemen les regles per a una adaptació en concret.

- *IDRULE* Identificador de regla
- *NAME* Nom de la regla
- *ACTIVE* Indicador de si la regla esta activa o no
- *CONTENT* La regla
- *IDADAP* Clau forànea a IDADAP de DADAP

DPROPSET
Taula on s'enmagatzemen les propostes generades. Esta relacionada amb DADAP mitjançant DADAPPROPSET.

- *IDPROPSET* Identificador de la proposta
- *CONSTRAINTS* Restriccions a complir
- *CHANGES* Canvis a aplicar
- *RAWPROPSET* Contingut de la proposta serialitzat